

Industrial^{IT} Compact Control Builder AC 800M Version 5.0 SP2

Getting Started Introduction and Installation



ABB

Industrial^{IT}
Compact Control Builder AC 800M
Version 5.0 SP2

Getting Started
Introduction and Installation

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without written permission from ABB, and the contents thereof must not be imparted to a third party nor used for any unauthorized purpose.

The software or hardware described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

This product meets the requirements specified in EMC Directive 89/336/EEC and in Low Voltage Directive 72/23/EEC.

Copyright © 2003-2008 by ABB.
All rights reserved.

Release: December 2008
Document number: 3BSE041584R5022

TRADEMARKS

All rights to trademarks reside with their respective owners.

TABLE OF CONTENTS

About This Book

| | |
|---|----|
| Document Conventions | 10 |
| Use of Warning, Caution, Information, and Tip Icons | 11 |
| Applicable Specifications | 11 |

Section 1 - Introduction

| | |
|--|----|
| Documentation Strategy | 13 |
| Product Overview | 13 |
| Project Explorer | 14 |
| Libraries | 15 |
| Supported PLC and Configurations | 16 |
| Compact Control Builder Functions | 17 |
| Functions from 800xA | 18 |
| Multi-User Engineering | 19 |
| Using Online Help in Compact Control Builder | 20 |
| Project Documentation | 20 |
| Online Manuals | 20 |
| Getting Help in Compact Control Builder | 21 |

Section 2 - Installing Software

| | |
|----------------------------------|----|
| Installation Prerequisites | 24 |
| Step-by-Step Instructions | 25 |
| Installing the Software | 25 |
| Starting Up | 27 |
| Control Builder | 27 |
| SoftController | 28 |
| OPC Server | 29 |
| Configuration Issues | 30 |

Section 3 - Compact Control Builder User Interface

| | |
|--|----|
| Introduction | 31 |
| About Programs and Projects | 31 |
| Project Templates | 32 |
| Project Explorer | 33 |
| Title Bar, Menu Bar, and Tool Bar..... | 33 |
| Project Explorer Pane | 34 |
| Libraries Folder..... | 36 |
| Applications Folder..... | 36 |
| Controllers Folder | 37 |
| Context Menus | 39 |
| Message Pane..... | 39 |
| Editors | 40 |

Section 4 - MyDoors Project

| | |
|---|----|
| MyDoors Project | 42 |
| Specifications | 42 |
| Defined Variables..... | 43 |
| Creating MyDoors Project | 44 |
| Local Variables | 46 |
| Function Blocks | 49 |
| Code Blocks | 51 |
| Code Input..... | 53 |
| Testing MyDoors Project | 60 |
| Project Examples in Control Builder | 65 |

Section 5 - Hardware Configuration

| | |
|---|----|
| Configure Hardware | 69 |
| Connect Variables to I/O Channels | 74 |
| Method 1 - Using Dot Notation | 74 |
| Method 2 - Using a Path Selector | 75 |
| Reading I/O addresses from the Application..... | 78 |

Section 6 - Connecting the PLC and Go Online

| | |
|--|----|
| Firmware Upgrade | 79 |
| Setting an IP Address | 82 |
| Setting IP Address for PLC | 82 |
| Setting IP Address for PC | 85 |
| Downloading the Project via Ethernet..... | 86 |
| Setting the System Identity in Control Builder | 86 |
| Downloading the Project to a PLC..... | 88 |
| Test the Program Online | 90 |

Appendix A - Compact Control Builder AC 800M Settings

| | |
|--|-----|
| Product Settings for Compact Control Builder | 94 |
| Memory Reservation | 94 |
| Language | 96 |
| File Locations | 97 |
| Multi-User Configuration | 98 |
| Creating a Shared Project Folder | 99 |
| Setting Up Compact Control Builder Stations | 100 |
| Setting Up OPC Server | 103 |
| Configuration Example | 108 |
| Guide lines for Multi-User Engineering | 110 |
| Download..... | 111 |
| General Download..... | 111 |
| Download New Project to Controller..... | 114 |
| Download Project to Selected Controllers | 114 |
| Disable/Enable Difference Report..... | 116 |

Appendix B - Network Redundancy

| | |
|---|-----|
| Setting Up Redundant Network..... | 119 |
| Two Separate Redundant Networks | 119 |
| Decide IP Addresses | 122 |
| Setup Using the IPConfig Tool | 123 |
| Configure PLC Ports from Project Explorer | 123 |

| | |
|--|-----|
| Configure PC Ports in Windows XP Professional | 124 |
| Download Project and Go Online | 125 |
| Setting Clock Synchronization using the CNCP Protocol..... | 125 |
| Design | 127 |
| IP Address..... | 127 |
| Separating Client/Server and Control Network | 129 |
| Summary of Configuration Steps..... | 131 |

Appendix C - Upgrade

| | |
|--|------------|
| Installation Considerations before Upgrade | 134 |
| Modifying Control Builder Installation | 134 |
| Introduction | 137 |
| Licenses..... | 137 |
| Products | 138 |
| Applications | 139 |
| Saving Application and Configuration Data | 139 |
| Remove Products | 141 |
| Install Products | 142 |
| Restore Application and Configuration Data..... | 142 |
| Handling a download..... | 145 |
| Compatibility Issues | 148 |

Appendix D - Communication Cables

| | |
|--|-----|
| Connecting Control Builder PC to a PLC | 171 |
|--|-----|

Appendix E - Programming Languages

| | |
|---------------|-----|
| General | 173 |
|---------------|-----|

Appendix F - Glossary

INDEX

About This Book

Welcome to Compact Control Builder AC 800M - an effective and easy to use programming tool. This manual helps the users to get start with the Compact Control Builder for the first time.

This manual contains main sections and appendixes. The appendixes contain examples of upgrading projects and details about multi-user engineering and network redundancy.

This manual is organized as follows:

[Section 1, Introduction](#), provides an overview of the Compact Control Builder.

[Section 2, Installing Software](#), describes the steps to install a single user configuration, that is, a Compact Control Builder and an OPC Server installed on the same PC.

[Section 3, Compact Control Builder User Interface](#), explains the Compact Control Builder and its core interface, the Project Explorer.

[Section 4, MyDoors Project](#), contains a small sample project that helps in familiarizing with the Control Builder environment.

[Section 5, Hardware Configuration](#), contains steps to add or remove hardware units from the tree structure in the Project Explorer.

[Section 6, Connecting the PLC and Go Online](#), contains the pre-requisites for connecting a PLC and downloading a project to go online.

Document Conventions

The following document conventions are used in this manual:

- The names of screen elements (for example, the title in the title bar of a window, the label for a field of a dialog box, and so on) are initially capitalized.
- Capital letters are used to name the keyboard key. For example, press the ENTER key.
- Lowercase letters are used for the name of a keyboard key that is not labeled on the keyboard. For example, the **space bar**, **comma key**, and so on.
- Press CTRL+C indicates that the user must hold down the CTRL key while pressing the C key (to copy a selected object in this case).
- Press ESC E C indicates that the user must press and release each key in sequence (to copy a selected object in this case).
- The names of push and toggle buttons are boldfaced. For example, click **OK**.
- The names of menus and menu items are boldfaced. For example, the **File** menu.
 - The following convention is used for menu operations: **MenuName > MenuItem > CascadedMenuItem**. For example: select **File > New > Type**.
 - The **Start** menu name always refers to the **Start** menu on the Windows Task Bar.
- System prompts/messages are shown in the Courier font, and user responses/input are in the boldfaced Courier font. For example, if the user enter a value out of range, the following message is displayed:

Entered value is not valid. The value must be 0 to 30.

Variables are shown using letters in *Italic style*.

MaxLimit

Use of Warning, Caution, Information, and Tip Icons

This publication includes **Warning**, **Caution**, and **Information** where appropriate to point out safety related or other important information. It also includes **Tip** to point out useful hints to the reader. The corresponding symbols should be interpreted as follows:



Electrical warning icon indicates the presence of a hazard which could result in *electrical shock*.



Warning icon indicates the presence of a hazard which could result in *personal injury*.



Caution icon indicates important information or warning related to the concept discussed in the text. It might indicate the presence of a hazard which could result in *corruption of software or damage to equipment/property*.



Information icon alerts the reader to pertinent facts and conditions.



Tip icon indicates advice on, for example, how to design the project or how to use a certain function.

Although **Warning** hazards are related to personal injury, and **Caution** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, users are expected to comply fully with all **Warning** and **Caution** notices.

Applicable Specifications

This product meets the requirements specified in EMC Directive 89/336/EEC and in Low Voltage Directive 72/23/EEC.

Section 1 Introduction

The Compact Control Builder contains type solutions that are used for simple logic control, device control, loop control, alarm handling, and so on, and packaged as standard libraries. Self-defined types from other projects can also be inserted into the current project. The Control Builder supports five different programming languages, namely, Function Block Diagram, Structured Text, Instruction List, Ladder Diagram, and Sequential Function Chart. These conform to IEC 61131-3 standard.

Documentation Strategy

This manual provides an introduction about the Compact Control Builder and provides instructions about installing and using the product, for the new users.

For information about online help in Control Builder, see [Getting Help in Compact Control Builder](#) on page 21.

Product Overview

Compact Control Builder is a fully integrated application that runs on Windows XP Professional and Windows 2003 Server. It provides tools for programming applications and configures hardware units from the AC 800M hardware family.

Besides the server requirements, the minimum software requirements are:

- Microsoft Word 2003 or Microsoft Word 2007.
- Adobe Acrobat Reader version 4.0 or later.

Microsoft Word is required for creating project documentation and Acrobat Reader is required to read the online manuals.

Project Explorer

The core user interface of the Compact Control Builder is called Project Explorer. The Project Explorer is used to create and build control projects. A project contains the entire configuration needed for an AC 800M based control solution, including control applications and hardware settings. Context menus are helpful while configuring hardware units or connecting parameters. Right-click an object to open its corresponding context menu.

Both the software (programs, functions, and so on) and the hardware (the actual hardware connected to the PLC) are modelled in a project. The relationships are shown in [Figure 1](#).

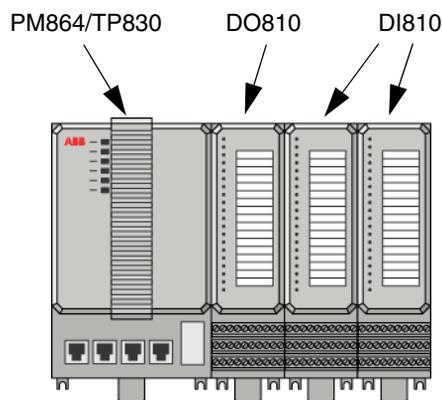
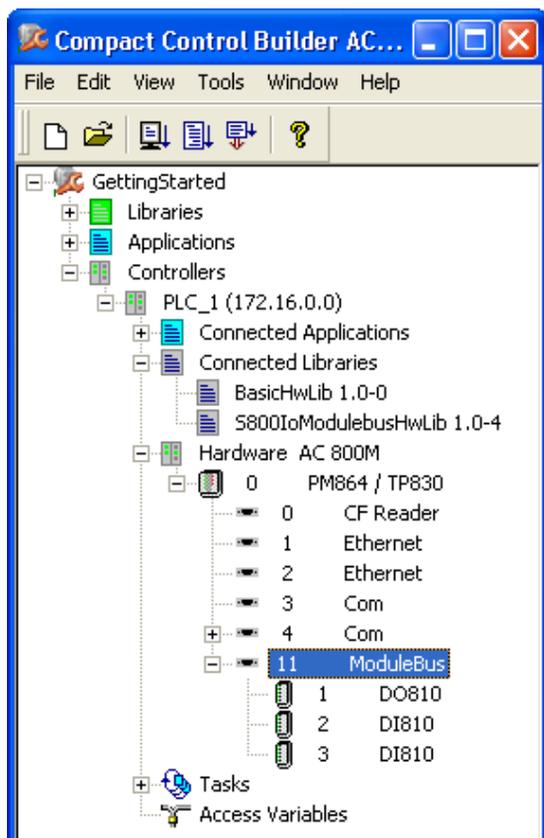


Figure 1. Project Explorer and actual hardware setup

Libraries

Compact Control Builder provides an extensive set of predefined type solutions stored in standard libraries. These include data types, functions, function blocks and Control Modules that can be used in the projects.

All standard libraries are included during the Control Builder installation and are available in the projects.

Compact Control Builder contains the following libraries:

- The Basic library contains basic building blocks for AC 800M control software like data types, function block types, and control module types, with extended functionality, designed by ABB.

The contents inside the Basic library can be categorized as follows: IEC 61131-3 Function Block Types, Other Function Block Types, and Control Module Types.
- The Communication Libraries include function blocks for MMS, ModBus, ModBus TCP, SattBus, COMLI and Siemens 3964R protocols.
- The Control Libraries include single PID control and cascade PID control function blocks, control modules, and so on.
- The Alarm and Event Library contains function blocks for alarm and event detection, and alarm printouts on a local printer.

Hardware

An extensive set of predefined hardware types, stored in standard hardware libraries, are delivered with Compact Control Builder. These hardware types are used in the projects when configuring the PLC hardware.

All the hardware types are included during the Compact Control Builder installation and are available in the projects.

The hardware types can be classified into the following:

- The Basic Hardware contains basic hardware types for PLC hardware, such as types for AC 800M, CPUs, Ethernet communication link, Com port, ModuleBus, and so on.

- The PROFIBUS Hardware contains hardware types for PROFIBUS communication interfaces, ABB Drives and ABB Panel 800.
- The Communication Hardware contains hardware types for the communication interfaces, MasterBus 300, ModBus TCP, IEC 61850, INSUM, DriveBus and RS-232C.
- Serial Communication Protocol Hardware contains hardware types for SerialProtocol, COMLI, ModBus and Siemens 3964R.
- The I/O System Hardware contains hardware types for I/O communication interfaces, I/O adapters and I/O units; S100 (incl. S100 Rack), S200, S800 and S900.

Supported PLC and Configurations

The AC 800M is the destination for the applications which are downloaded to the PLC from the Project Explorer. The programming code is then executed in the PLC.

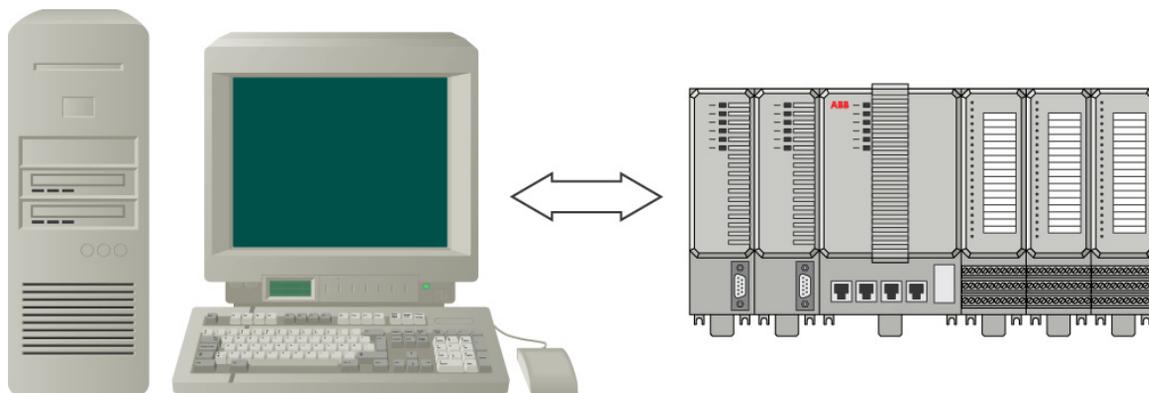


Figure 2. The Compact Control Builder station communicates with a PLC



Do not run more than one Compact Control Builder simultaneously on a PC.

Compact Control Builder Functions

The Compact Control builder is used to create control solutions. These solutions are created within control builder projects.

Several levels of structuring are available inside one project. A project in the Control Builder can handle up to 256 applications, and each application can handle up to 64 programs.

A maximum of 32 Control Builder PCs can be used together in multi-user environment, and a maximum of 32 PLCs can be created and handled within a project.

Using the Control Builder, self-defined libraries containing data types and function block types can be created in any project.

Besides the function block types, the Control Builder also handles control modules, which are components for object-oriented (and graphical) programming.

[Table 1](#) lists the main Compact Control Builder functions.

Table 1. Main Control Builder Functions

| Functions |
|--|
| Backup/Restore |
| Create/change/insert libraries |
| Create/change/use Control Modules |
| Difference report (between previous/new application) |
| Distribute code in an application to several PLCs |
| Downloading projects and go online |
| Multi-user engineering |
| Search and Navigation Tool |
| Testing projects off line |

Functions from 800xA

Additional functionality for building DCS type of control solutions can be used from the control builder available in the ABB 800xA DCS system.

The 800xA control builder (the CBM Professional) adds the following functions to the set of functions available in Compact Control Builder:

- Online Upgrade.
- Load Evaluate Go.
- Batch handling.
- Audit Trail.
- SFC Viewer.
- High Integrity Controller for SIL applications.
- CI860 for FF HSE, and CI862 for TRIO I/O.
- Information routing via HART protocol.
- Security (see [Appendix F, Glossary](#)).



The additional functions from 800xA are not included in the Compact Control Builder AC 800M.



If the additional license is purchased, Compact Control Builder solutions can be moved to 800xA systems, and the PLC projects can be opened in the Control Builder Professional.

Multi-User Engineering

Compact Control Builder supports multi-user engineering with a maximum of 32 separate engineering workplaces. In a multi-user configuration, all Control Builder PCs and the OPC Server must have access to the common project file(s). This means that a common Project folder must be created on a shared network server.

The network server can be placed anywhere in the network; in a Control Builder PC, in an OPC Server PC, or located as a stand-alone file server.

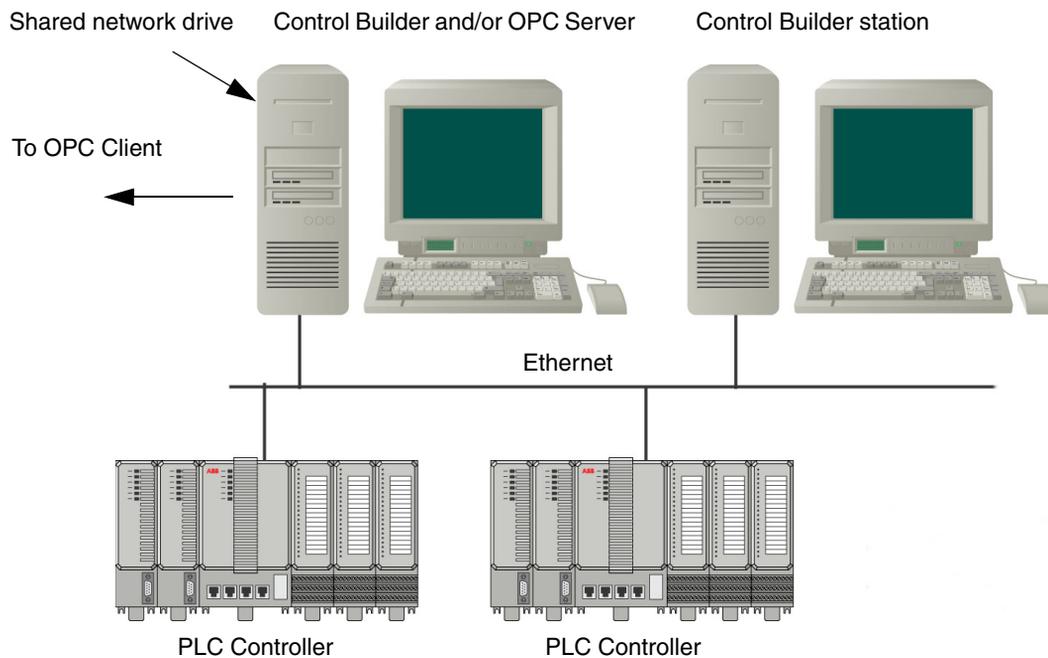


Figure 3. Programmers can share the same project. Multi-user engineering stores projects on a shared network drive

Using Online Help in Compact Control Builder

The Compact Control Builder provides online documentation.

Project Documentation

Compact Control Builder facilitates a project documentation feature for libraries, applications, and PLCs or for single types. The project documentation is developed as a Microsoft Word file and can be connected to either a standard document template or user/company specific template. A table of contents is automatically generated for every project document.

Creating Project Documentation

To create project documentation from the Project Explorer:

1. Right-click an object in the Project Explorer tree, and select **Documentation** to open the Documentation window.
2. Click **More** to open the Editor Properties window.
3. Select a tab in the window.
4. Click **OK**.

Online Manuals

The Compact Control Builder provides related manuals online.



Acrobat Reader is required to open and read the online manuals provided by the Compact Control Builder.

Accessing Online Manuals

To access the online manuals from the Project Explorer, select **Help > Manuals**.

Getting Help in Compact Control Builder

The Online help in the Compact Control Builder can be obtained by:

- Context-Sensitive Help (F1).
- Contents Topic.
- Index.
- Keyword Search.

Accessing context-sensitive Help

To access context-sensitive help for items in Project Explorer:

1. Select the element for which help is required (any item from the tree, command inside an editor, and so on).
2. Press the F1 key.

Accessing contents topic

Click Help in any pop-up window to view the Online help based on topics.

Using the Online Help Index

The index offers many ways to find the information:

- Enter the action about which the information is needed (for example, “configure” or “download”).
- Enter the name of the object about which the information is needed (for example, “PM864” or “project explorer”).



It is not always possible to find information about a single object by entering its name. Enter the name of the category instead (for example, “I/O units” or “data types”). This lists the objects or units, from which the topic can be selected.

- Enter the subject about which information is needed (for example, “function block types” or “communication interfaces”).



For information about a specific library object or a specific hardware unit, select the object in Project Explorer, and press F1 key.

Text Search

The text search runs through all topics and finds all matches. The text must be specific, else the search ends with too many search hits.

Section 2 Installing Software

This section explains how to install and start up a single-user configuration, which means a Compact Control Builder and an OPC Server installed together on the same PC station. The software delivered on the CD consists of two parts - the Compact Control Builder AC 800M and the OPC Server for AC 800M. Each of these is installed with the help of installation wizards.

- The first installation wizard installs Compact Control Builder, Base Software for SoftController, and User Documentation.
- The second installation wizard installs OPC Server for AC 800M.



Run the Compact Control Builder installation before running the OPC Server installation.



The Compact Control Builder opens the projects stored in a project folder created during the installation.

If the project folder path is changed in a recent installation, the previous projects cannot be found by the Control Builder. This problem is solved by either changing the project folder path back to previous location, or copying the previous projects from the Windows explorer into the current Project folder location.

Installation Prerequisites

The following prerequisites must be met before installing the software:

- Install a PC with either Windows XP Professional or Windows 2003 Server. For PC requirements, see [Product Overview](#) on page 13.
- Login to Windows with Administrator privilege.
- Remove previous Control Builder versions from the PC¹. This also includes other products that comes with a Compact Control Builder installation (for example, OPC Server for AC 800M).



Do not install Compact Control Builder on a PC that already has Control Builder Professional installed. A Compact Control Builder and a Control Builder Professional cannot coexist in a PC.

1. From Windows Control Panel select **Start > Control Panel**. Select **Add or Remove Programs** from the list.

Step-by-Step Instructions

Install the software from the CD onto the local disk, as the software cannot be run from the CD or a network drive.

Installing the Software

1. Login as Administrator in Windows.
2. Insert the CD into the drive. After a few seconds the Welcome dialog appears as shown in [Figure 4](#). If the dialog does not appear, start the file *Startme.bat*, located in the root directory of the CD.

Compact Products for AC 800M - Installation



Figure 4. The Welcome dialog of the installation process

The installation dialog contains the following buttons:

- The **Release Notes** button provides the latest information.
- The **Install Software** button activates the installation procedure.
- The **Installation help** button accesses information on how to install a product.
- The **Exit** button quits the installation procedure.

Installing the Compact Control Builder

1. Click **Install software** in the Installation dialog, to open the software installation window.



Always start with the Compact Control Builder installation. The OPC Server installation needs to read the Control Builder settings that are created during the Compact Control Builder installation.

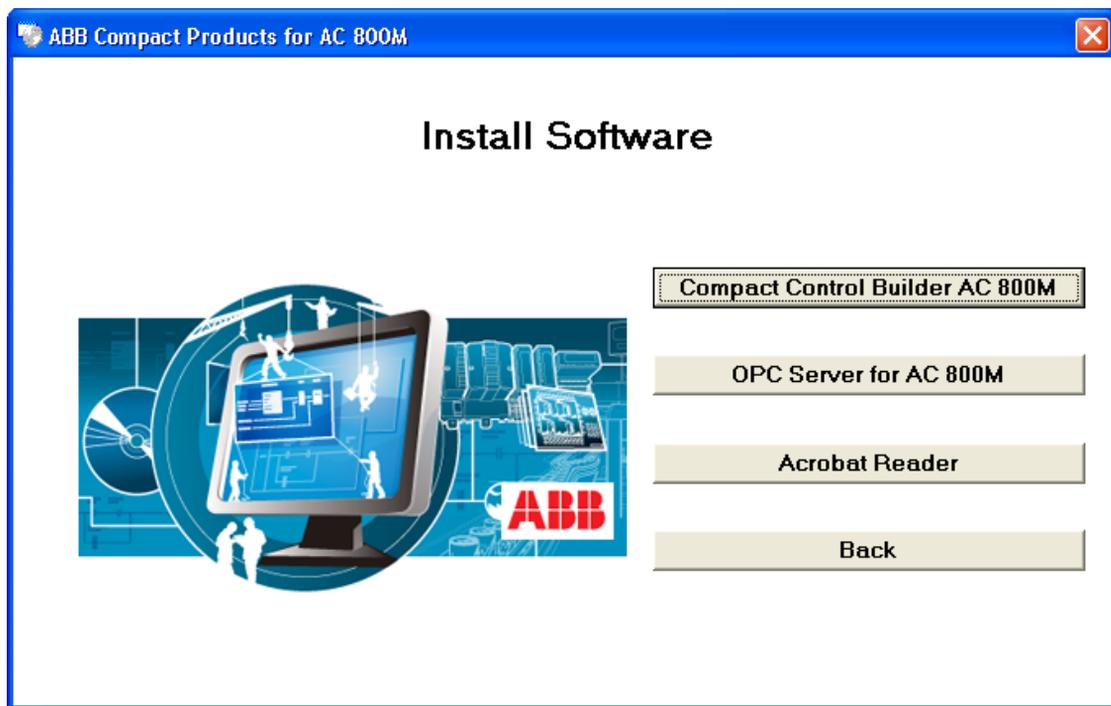


Figure 5. The Software Installation dialog

2. Click **Compact Control Builder AC800M** to start the Installation Wizard.
3. Follow the on-screen installation instructions.



Clicking **Cancel** in any of the installation wizard dialogs interrupts the installation. When installation procedure is interrupted, all the previously installed components are disregarded.

Installing the OPC Server for AC 800M

1. Click **OPC Server for AC 800M** in the Install Software dialog, to start the Installation Wizard.
2. Follow the on-screen installation instructions.

Running the OPC Server on the same PC as the Compact Control Builder does not require further settings.



For more information about setting up an OPC Server for multi-user engineering, see [Setting Up OPC Server](#) on page 103.

Starting Up

Control Builder

Starting the Compact Control Builder

Double-click the Control Builder icon on the desktop (if selected during installation), or from the Start menu on the Windows Task Bar, **Start > All Programs > ABB Industrial IT AC 800M > Compact Control Builder AC 800M**.

SoftController

The SoftController is a simulation tool that runs with Base Software. A SoftController allows the download of projects from the Project Explorer even though the user may not have access to a real AC 800M controller. Instead of downloading to a PLC, the projects can be downloaded to the SoftController.



In order to start the SoftController, it is necessary either to have administrator privileges in Windows, or be part of the local group *ABB Controller user group*. Contact the administrator and apply to be a member of this group. The *ABB Controller user group* is created automatically in Windows during the SoftController installation.

Starting the SoftController

The following steps help to start the SoftController and to locate its network address (the address must be set in Project Explorer and OPC Server panel).

1. Double-click the SoftController icon on the desktop (if desktop shortcut is selected during installation), or start the SoftController from the Start menu:

Start > Programs > ABB Industrial IT AC 800M > > SoftController 5.0.

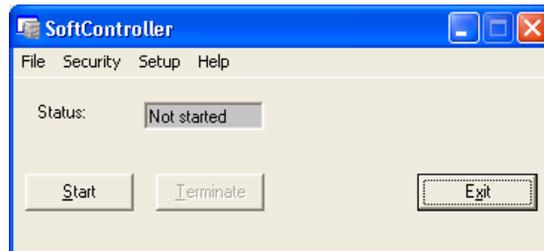


Figure 6. The SoftController start panel

2. Click **Start**. The Status field displays *Started* and the SoftController starts.
3. Select **File > View log file**. A Session.LOG file opens in Notepad.
4. Scroll down to find a network address. An example of an IP address:
10.46.35.117:2.



To find out the IP address for a PC, open the command prompt (DOS editor) and run the command **ipconfig**.

5. Close the Notepad program.

To learn more about running an application in a SoftController, see [Section 5, Hardware Configuration](#) and [Section 6, Connecting the PLC and Go Online](#).

Stopping the SoftController

1. Click **Terminate** to open the SoftController dialog.
2. Click **Yes**. The Status field displays *Not started* and the SoftController stops.
3. Click **Exit**.

OPC Server

After the OPC Server is installed, it is easy to connect a PLC to the OPC Server from the OPC panel. Before doing this, ensure that the controller is connected to the network.

Starting the OPC Server

Double-click the OPC Server icon on the desktop (if desktop shortcut is selected during installation), or start the OPC Server from the Start menu:

**Start > Programs > ABB Industrial IT AC 800M > >
OPC Server for AC 800M 5.0.**

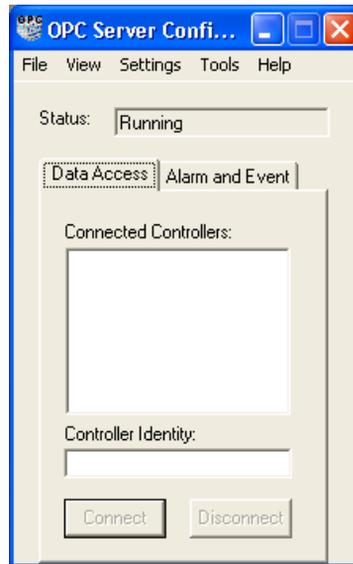


Figure 7. The OPC Server Configuration window

Connecting the OPC Server

1. Open the Data Access tab screen.
2. In the Controller Identity field, enter the IP address of the PLC.
3. Click **Connect**.
4. Open the Alarm and Event tab screen, and repeat Step 2 and Step 3.



For information on creating a PLC Id, see [Setting an IP Address](#) on page 82 and [Setting the System Identity in Control Builder](#) on page 86.

Configuration Issues

The Compact Control Builder supports multi-user engineering. For more information see [Appendix A, Compact Control Builder AC 800M Settings](#).

This manual also covers configuration issues like network redundancy and upgrading projects to 5.0 versions. For more information, see [Appendix B, Network Redundancy](#) and [Appendix C, Upgrade](#).

Section 3 Compact Control Builder User Interface

Introduction

This subsection provides a brief introduction to the Compact Control Builder, and its core interface, the Project Explorer.

About Programs and Projects

The following list describes the hierarchy among projects, applications, programs, and tasks in the Compact Control Builder.

- A project is the top level software unit and it contains the configuration data for libraries, applications, connected hardware, and so on. It also groups libraries, applications and the connected hardware in an hierarchical tree structure in the Project Explorer.
- Each application contains programs and additional objects (data types, function block types, control module types) that are used within the application.
- Each program is connected to a task, which decides how often the program is executed. It is also possible to connect individual function blocks and control modules to different tasks.

[Figure 8](#) shows the sequence from creating a new project to performing a download.

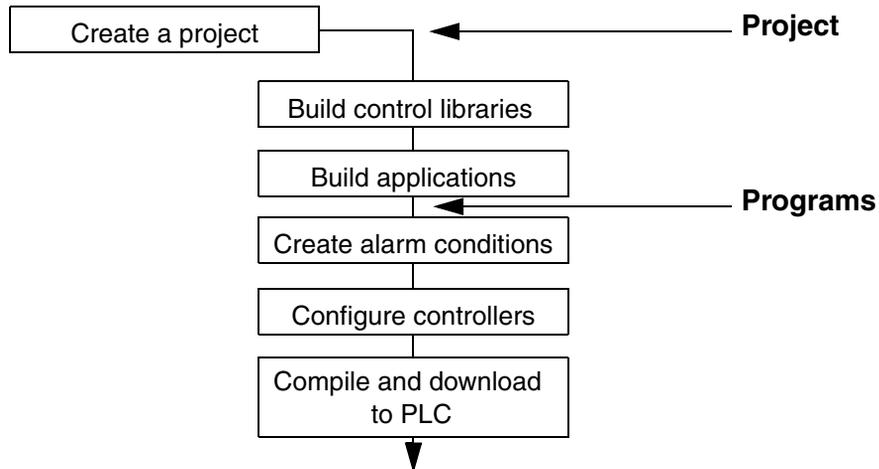


Figure 8. Sequence for building a project and the hierarchy between a Project and Programs

Project Templates

While creating a new project, the Control Builder provides a set of predefined templates, typical for a control system. The templates contain predefined initial setup data that is suitable for different kind of projects. The Compact Control Builder provides the following project templates:

- AC800M, for normal use.
- SoftController, for development use, without access to a PLC.
- EmptyProject, consisting a minimum configuration with only the System folder inserted.

An empty project template contains only the mandatory system firmware functions, with no additional application or hardware functions.

Project Explorer

Project Explorer is the core user interface in the Control Builder programming tool. It displays the currently active project. Only one project can be open at the same time.

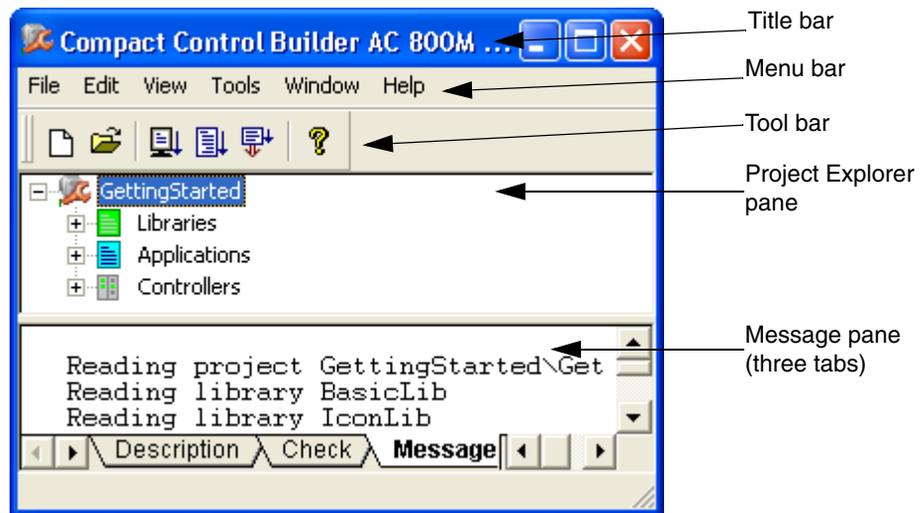


Figure 9. Project Explorer

Title Bar, Menu Bar, and Tool Bar

The title bar shows the project name.

The menu bar contains the drop-down menus: File, Edit, View, Tools, Window, and Help.



When menu items on the menus are grayed out, they cannot be accessed (the function is not allowed in the current context).

The tool bar contains icons that serve as shortcuts to the most common Control Builder functions, such as online help and download.



For detailed information about the menu bar and tool bar, refer to the Control Builder online help.

Project Explorer Pane

The Project Explorer pane contains three main folders, see [Figure 10](#):

- The Libraries folder, see [Libraries Folder](#) on page 36.
- The Applications folder, see [Applications Folder](#) on page 36.
- The Controllers folder, see [Controllers Folder](#) on page 37.

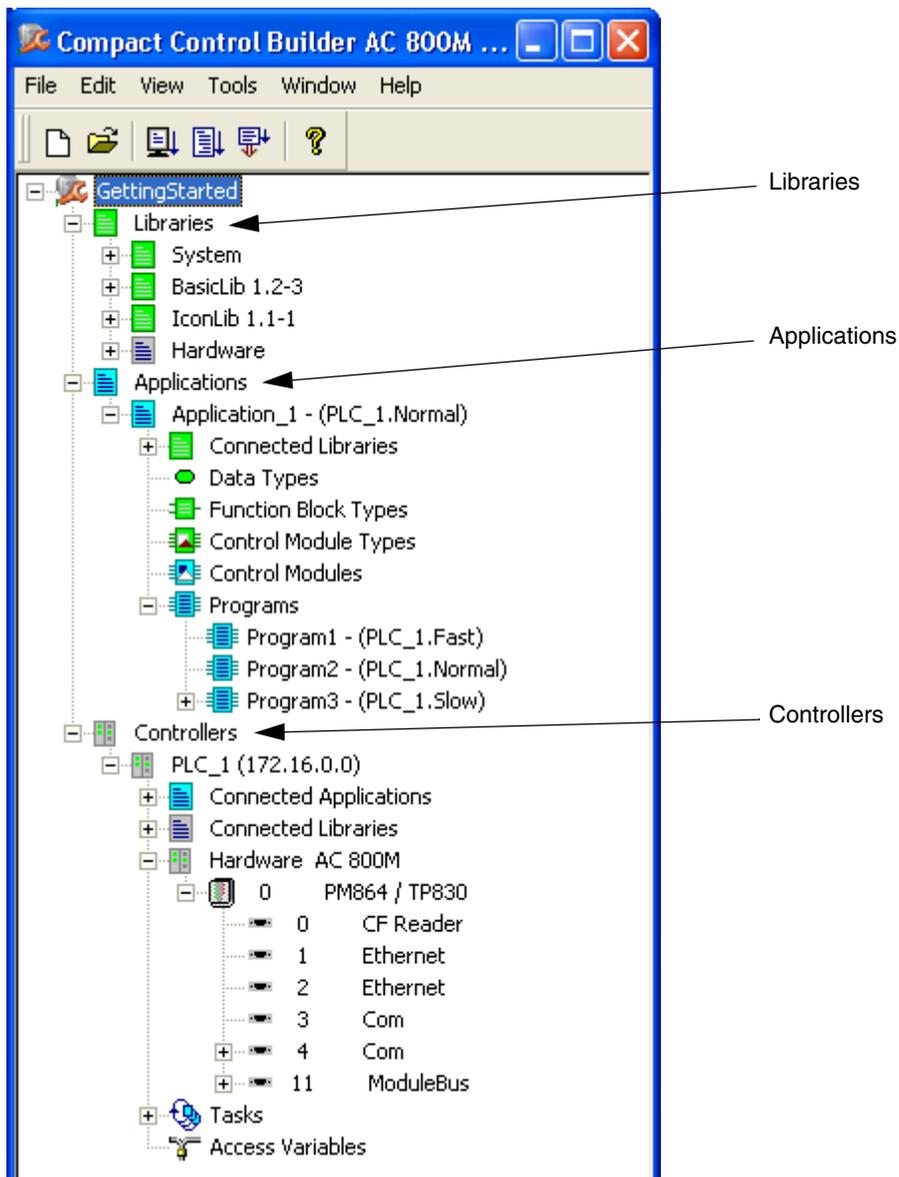


Figure 10. The Project Explorer pane, showing the three main folders, Libraries, Applications, and Controllers

Libraries Folder

When a project is created, the Libraries folder contains the System folder (containing firmware functions that can be used throughout the applications) and the Basic and the Icon libraries (the two libraries that are always connected to a project).

Besides these three, the Libraries folder also contains a Hardware folder with a sub-folder containing Basic hardware types (BasicHWLib).

When the project is created, both the standard libraries and self-defined libraries can be added into the Libraries and Hardware folders.



A library can only be added to an application if it has first been added to the Libraries folder. A hardware unit (type) can only be connected in a controller configuration if the corresponding hardware library has first been added to the Hardware folder.



For more information on libraries and library handling, see Online Help.

Applications Folder

The Applications folder holds all the code that is downloaded to the PLC(s). This code can be stored either as program, or as control module types or single control modules. The choice depends on the requirements of the particular application.

The Connected Libraries folder contains all libraries that are connected to this particular application. Libraries are connected by right-clicking this folder and selecting Connect Library. However, only libraries that have already been inserted to the project can be connected to an application. In order to access the types inside a library, it must be connected to the application. Connect a library to an application by right-clicking the Connected Libraries icon and select a library from the drop-down menu.

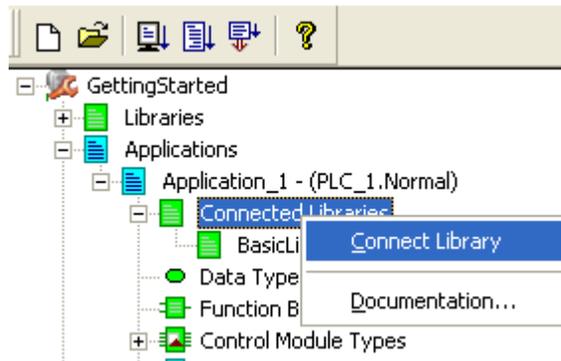


Figure 11. The context menu for connecting a library to an application

The Application folder contains three sub-folders for collecting types: Data Types, Function Block Types, and Control Module Types. The user can either insert an existing type (from another project) or create a new type within any of these three sub-folders.

There are two ways of organizing the code, in programs or in control modules. Control modules are stored in the Control Modules folder, while programs are stored in the Programs folder. By default, the Programs folder contains three programs. Each of these three programs are connected to a default task. The task connections can be changed, as well as the own tasks and programs can be added.



From the objects in the Applications folder, a number of software editors can be opened, see [Editors](#) on page 40.

The code can be checked for errors by clicking on the Check icon on the tool bar. Errors are indicated by a red triangle next to the object in question (in offline mode). Descriptions of the errors (if any) are displayed in the Check tab of the message pane.

Controllers Folder

The Controllers folder contains all controllers that belong to the project.

Each controller has a Connected Applications folder with the application(s) running in the controller, and a Connected Hardware Libraries folder with all hardware types to be used when configuring the controller. The applications and Hardware Libraries are connected by right-clicking the folder and selecting **Connected Applications**

and **Connected Hardware Libraries** respectively. It is only hardware libraries added to the project that can be connected to a controller.

For each controller, there is a CPU unit to which other hardware units, such as I/O units and communication interfaces can be added. Units can also be added to the controller on the same level as the CPU unit. The controller structure mirrors the physical structure, which means that all ports and buses have their own corresponding unit (icon) in Project Explorer.



For more information about hardware configuration and the Controllers folder, see [Configure Hardware](#) on page 69.

The Controllers folder also contains a sub-folder Tasks. The tasks folder contains tasks that are used to control the execution of the applications. By default, the Tasks folder contains three tasks: Fast, Normal, and Slow. However, the tasks can be added to the applications as needed. The Connected Applications folder contains all the applications that are connected to the PLC.

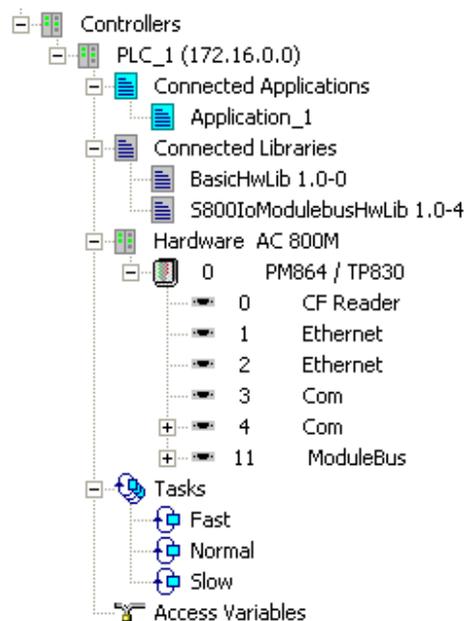


Figure 12. The PLC structure in Project Explorer, with corresponding icons for ports and buses

Double-click the ‘Tasks’ folder to open a task overview. Double-clicking an individual task, displays the Task Properties dialog for that particular task.



CPU units, I/O units, communication ports, communication interfaces, and so on, can be opened using editors, see [Editors](#) on page 40.

Context Menus

Context menus can be used to edit the properties of various objects. Context menus are displayed by right-clicking an object in Project Explorer.

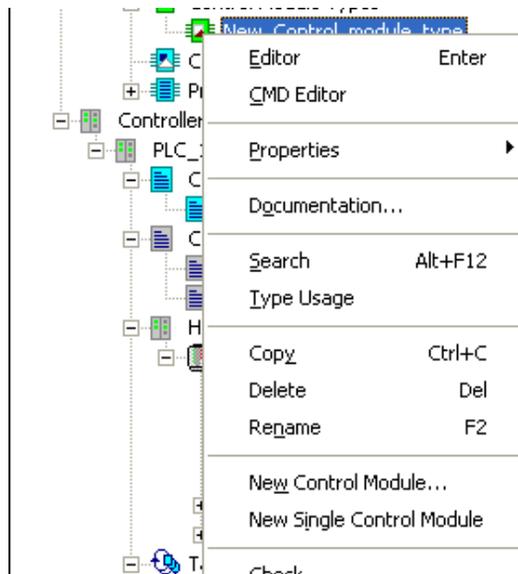


Figure 13. Context menu in Project Explorer

Message Pane

The message pane contains three tabs:

- **Description**, shows a description of the selected type or hardware object.
- **Check**, shows the result of a code check, including error messages.
- **Message**, showing messages resulting from events in Control Builder, such as compiling and loading a new project.

Editors

Control Builder contains a number of editors. The editors can be accessed from Project Explorer. To access an editor, right-click on the object (a PLC, another hardware unit, an application, a program, or a type) and select the editor from the context menu.

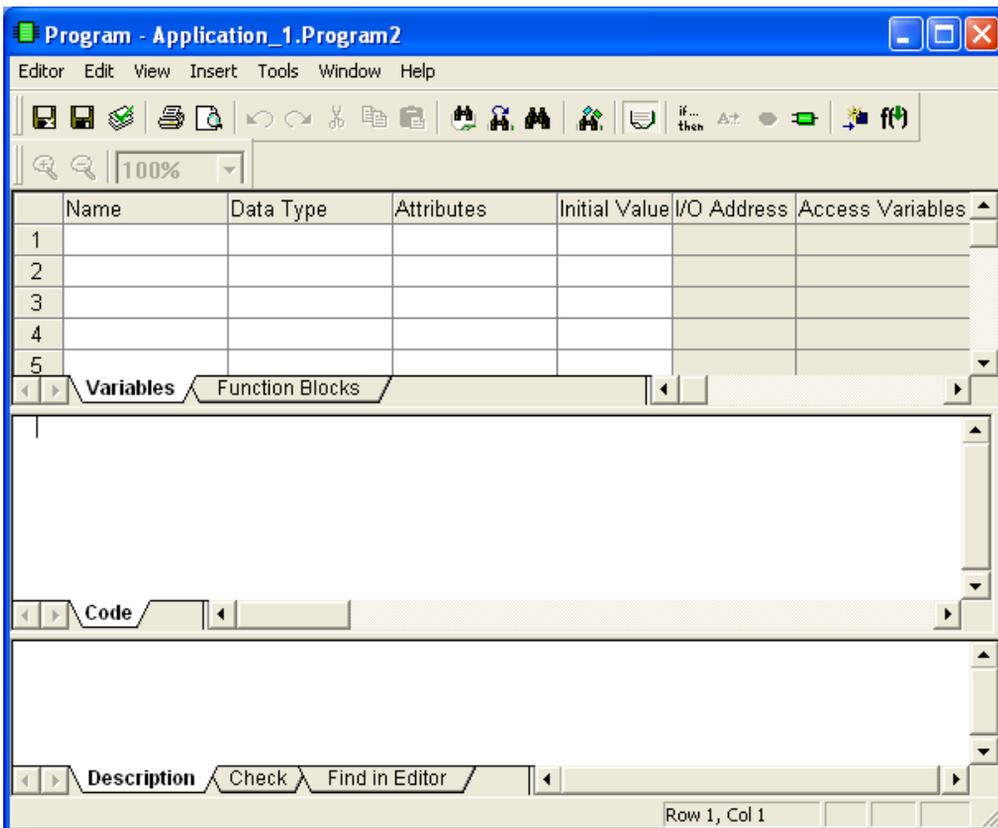


Figure 14. Program editor

Among many things, editors are used to declare project constants, and parameters, as well as to declare variables and connect them to I/O channels. There are also a number of programming language-specific editors, such as the Function Block Diagram (FBD) editor and the Control Module Diagram (CMD) editor.

Section 4 MyDoors Project

This section describes building a small project and getting familiarized with the Control Builder environment. An example project called MyDoors, to simulate the entrance to a store is created. While working with the MyDoors project, the concept of declaring variables, function blocks, and separating code by using code blocks, is also described.

At the end of the MyDoors project, the application is tested in the Control Builder Test mode. This helps to verify, in a secure way, how variable values and conditions are changing during a program execution.



If the access to a PLC or IO modules is not available, build this project with a SoftController. Look for SoftController specific instructions throughout the MyDoors project example.

However, to study the MyDoors project example, the Control Builder installation comes with an existing example called ShopDoors. See the subsection [Project Examples in Control Builder](#) on page 65, for locating the project examples.



For locating the ShopDoors example, or any other Control Builder examples, follow the instruction given under section [Project Examples in Control Builder](#) on page 65.

Once the guidelines are followed in this section, proceed with the further sections:

- [Section 5, Hardware Configuration](#) explains how to setup a hardware configuration according to the control system.
- [Section 6, Connecting the PLC and Go Online](#) covers all the essential steps for making it possible to download an application and go online.

MyDoors Project

Before creating the project and writing the code, refer to the [Specifications](#) on page 42 and the [Defined Variables](#) on page 43.

Specifications

This project simulates the entrance to a store. The following specifications are given:

- The entrance consists of two sliding doors that open when a customer activates a photocell.
- Each door is opened and closed by its own motor.
- The doors return to default position (closed) five seconds after the last activation of the photocell. Consequently, several customers arriving one after the other extends the time the door remains open.
- The number of customers is recorded for statistics. Manual reset of this counter should be possible.
- The total number of times the doors have opened since they were last serviced should be recorded.
- Each time the door opens, the counter should be incremented. When the counter reaches a preset limit, a flag indicates that service is required. Manual reset of the flag is possible.

Defined Variables

- **Photocell**

The photocell has two states, active and inactive, typically represented by a Boolean variable. In this project, a Boolean variable named `Photo_Cell` (true = active, false = inactive) is used.
- **Door motors**

The entrance itself consists of two doors facing each other. Each door is opened by a motor controlled by Boolean signals (`Motor_1` and `Motor_2`). The time the doors should remain open is declared in a variable `DoorsOpen_Time` of type `time`.
- **Number of customers**

Each time the photocell is activated, a counter representing the total number of customers entering the shop should be incremented. The counter, `Customers_Qty`, is of type `integer`.
- **Reset the counter on certain dates**

On certain dates, the shop manager records the total number of customers up to that date, and resets the counter. Consequently, a Boolean variable `Reset_Counter` is declared, which resets the counter.
- **Door service intervals**

The doors should have regular service intervals, approximately after every 10,000 openings; also the number of openings from the previous service need to be recorded. The record is represented as the variable `Openings_Freq` of type `dint`.
- **Time for service**

When the counter reaches the upper limit defined by `Openings_Total` of type `dint`, a flag (`Service_Req` of type `Boolean`) is set, indicating that service is required. Manual reset of the service counter is activated using a Boolean variable `Serviced`. The doors should continue to work even if service is not performed.

Creating MyDoors Project

Creating a New Project

1. From the Project Explorer, select **File > New > Project**. A New Project window opens.

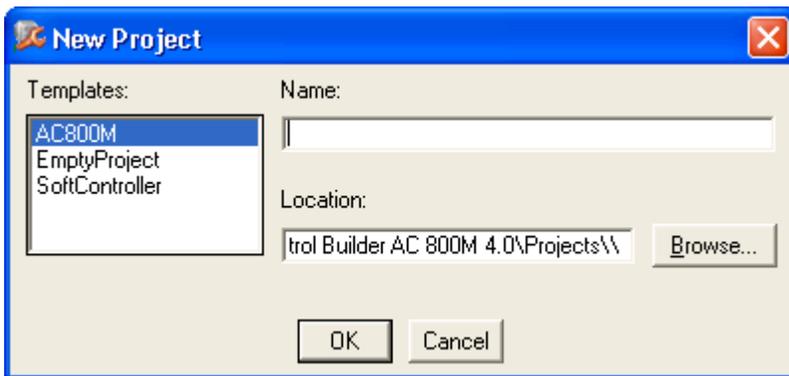


Figure 15. The New Project window for setting up a project

2. Select the **AC800M** template and type *MyDoors* in the **Name** field. Ignore the given location path (for now).
3. Click **OK**. Project Explorer creates and opens MyDoors project, see [Figure 16](#).

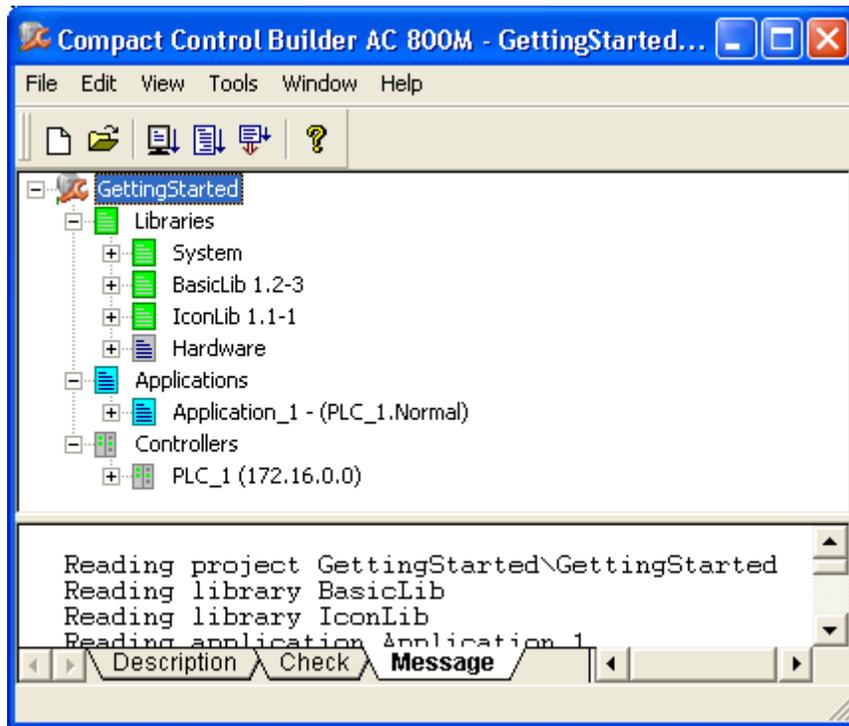


Figure 16. MyDoors project opened in Project Explorer

The Libraries folder contains the standard libraries Basic library (BasicLib) and Icon library (IconLib).



The System folder is always automatically inserted into a project. It contains firmware functions and cannot be removed from the project or changed by the user.

Local Variables

There are different types of variables in Compact Control Builder for storing and computing values (local, global, and access variables), where the local variables are the most frequently used in Control Builder. They always belong to the local code inside a function block, control module or a program.

This example describes how to declare local variables in a program named **Program2**.

Declaring Local Variables

1. In the Project Explorer, expand the project tree to see **Program2**, (Figure 17). Double-click the icon to open the Program editor.

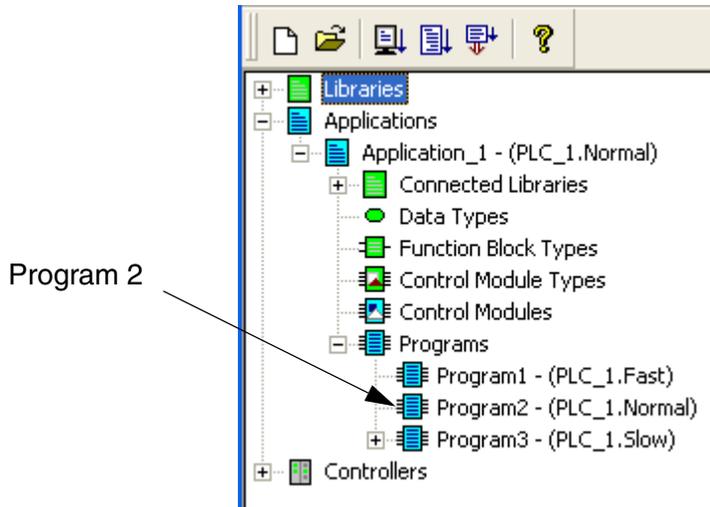


Figure 17. Programs folder expanded with three preset programs available

2. The program editor is divided into three panes: the declaration pane, the code pane, and the message pane, see Figure 18.

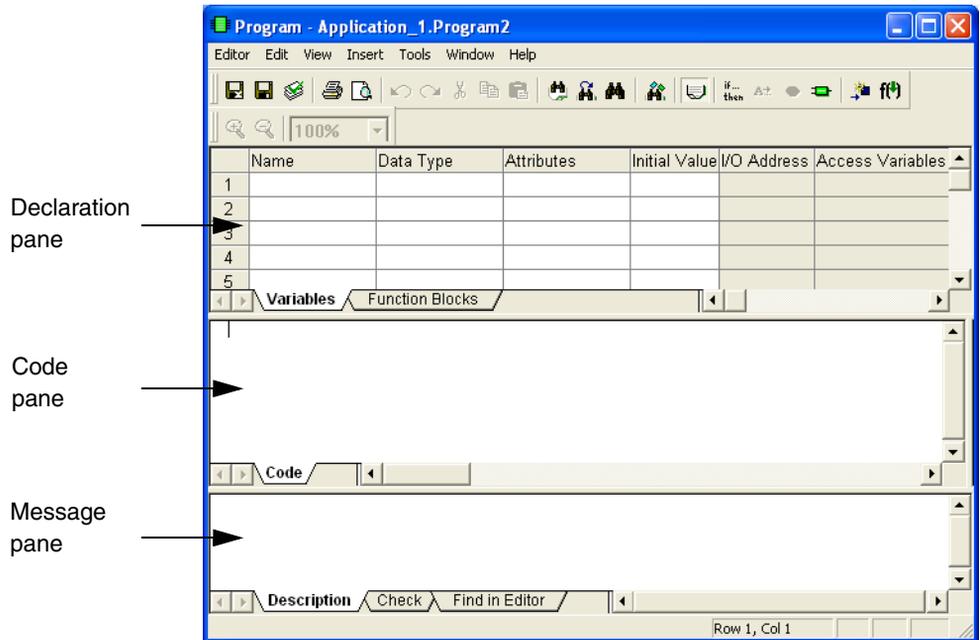


Figure 18. The editor for program **Program2**

3. Place the cursor in the upper left-hand cell in the declaration pane and enter `Photo_Cell` in the 'Name' column.
4. Move one cell to the right by pressing the tab key. Type `bool` in the "Data type" column. Move the cursor to next column labeled "Attributes".
5. Choose the default setting `retain` (which means that the variable keeps its value at a restart). Press the tab key to move to the next column.
6. Set the initial value to `false` to indicate that the doors are closed at start-up.
7. Skip the column I/O address. The address is automatically filled in later when configuring the hardware.
8. Description can be entered in the last column. The declaration of the Boolean variable is shown in [Figure 19](#).

| | Name | Data Type | Attributes | Initial Value | I/O Address | Access Variables | Description |
|---|------------|-----------|------------|---------------|-------------|------------------|---------------------|
| 1 | Photo_Cell | bool | retain | false | | | Photo cell to doors |

Figure 19. Declaration of the Boolean variable `Photo_Cell`

9. Declare a second variable named, `DoorsOpen_Time` which represents duration of time the doors remain open. Complete the declaration of `DoorsOpen_Time` according to row 2 in [Figure 20](#).

| | Name | Data Type | Attributes | Initial Value | I/O Address | Access Variables | Description |
|---|----------------|-----------|------------|---------------|-------------|------------------|--|
| 1 | Photo_Cell | bool | retain | false | | | Photo cell to doors |
| 2 | DoorsOpen_Time | time | constant | T#5s | | | Time duration that doors should be opened |
| 3 | DoorsOpen_ET | time | retain | | | | Elapsed time after photo cell has been activated |

Figure 20. Declaration of the `DoorsOpen_Time` and `DoorsOpen_ET` variables



For the attribute constant of the variable `DoorsOpen_Time`, either explicitly enter **constant**, or scroll through the available formats using Alt-key together with the up and down arrow keys.

10. Declare a variable named, `DoorsOpen_ET` that records the time elapsed since the photocell was activated last time. Complete the declaration of `DoorsOpen_ET` as shown in row 3 in [Figure 20](#).
11. Declare the remaining variables (starting from row 4) in the grid as shown in [Figure 21](#).

| | | | | | | | |
|----|----------------|------|----------|-------|--|--|--|
| 3 | DoorsOpen_ET | time | retain | | | | Elapsed time after photo cell has been activated |
| 4 | Motor_1 | bool | retain | false | | | Output to motor opening door 1 |
| 5 | Motor_2 | bool | retain | false | | | Output to motor opening door 2 |
| 6 | Openings_Total | dint | constant | 10 | | | Total number of openings until service |
| 7 | Openings_Freq | dint | retain | | | | Number of openings since last service |
| 8 | Serviced | bool | retain | false | | | Flag that resets the number of openings |
| 9 | Service_Req | bool | retain | false | | | Flag that is set when service is required |
| 10 | Customers_Qty | dint | retain | 0 | | | Total number of customers |
| 11 | Reset_Counter | bool | retain | false | | | Flag that resets the number of customers |

Figure 21. Declaration of the remaining variables



To specify the Data types in the column, choose **Insert > Variable, Type, Attribute** from the editor (or press CTRL+J) to access a list of possible data types.

12. Click **Check**  to check for errors.
13. Click **Save**  to save the variables.

Function Blocks

Timers and counters in the Compact Control Builder are normally represented as function block types and located in the Basic library. This example declares one Timer (TOF), and two Counters (CTU) from the Basic library.

Declaring Function Blocks

Make sure the program editor is open, (see [Figure 17](#), to open editor)

1. Select the **Function Blocks** tab in the declaration pane.
2. Place the cursor in the upper left-hand cell in the declaration pane and type `OpenDoors` in the Name column.
3. Move one cell to the right by pressing the tab key. Right-click the cell and select **Insert > Variable, Type, Attribute** from the context menu. A dialog list opens.

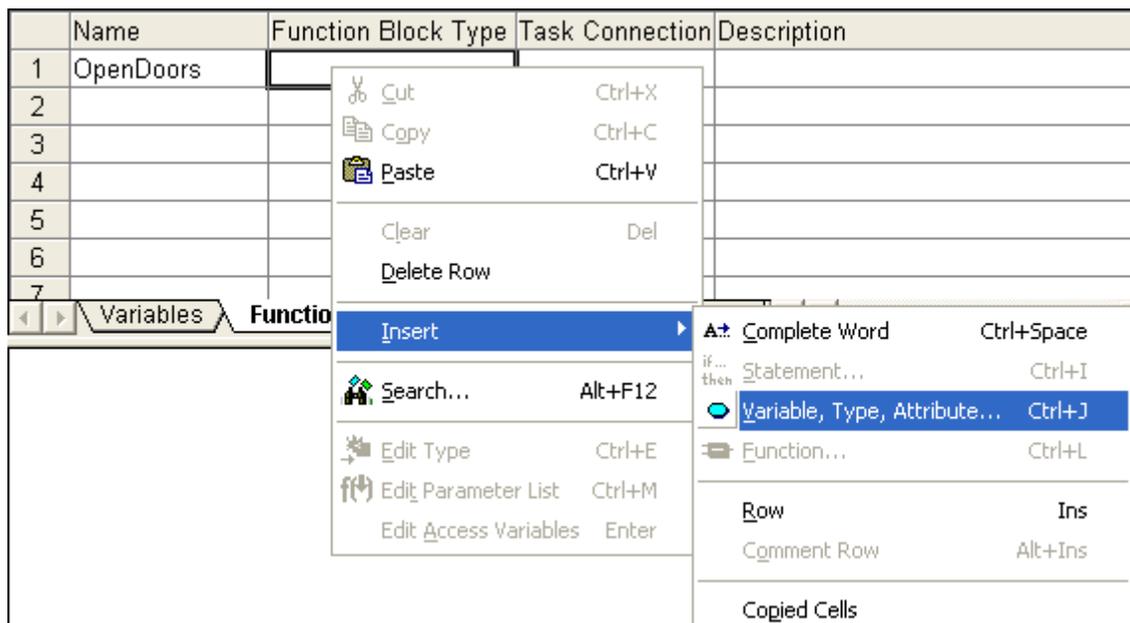


Figure 22. The path in the context menu for selecting (for example) function blocks

- Type **TO** (or TOf) to jump down to the TOf Function Block type in the list.

| | Name | Function Block Type | Task Connection | Description |
|---|-----------|---------------------|-----------------|-------------|
| 1 | OpenDoors | TOf | | |
| 2 | | TimerU | | |
| 3 | | TOf | | |
| 4 | | TON | | |
| 5 | | TP | | |
| 6 | | Trigger | | |
| 7 | | BasicLib 1.2-3 | | |

Variables | **Function Blocks**

Figure 23. The TOf function block type is selected

- Press the ENTER key to declare the TOf in the program editor. Write Description text according to Figure 24.
- Similarly, declare the two CTU function blocks in row 2 and row 3. Name them Customer_Count_Up and Service_Count_Doors, according to Figure 24.

| | Name | Function Block Type | Task Connection | Description |
|---|---------------------|---------------------|-----------------|--|
| 1 | OpenDoors | TOf | | Timer for motor |
| 2 | Customer_Count_Up | CTU | | Counter for the number of customers |
| 3 | Service_Count_Doors | CTU | | Counter for the number of openings o doors |
| 4 | | | | |

Variables | **Function Blocks**

Figure 24. Declaring the function blocks



For information about the TOf and the CTU function blocks, refer to the Control Builder Online Help. Place the cursor in the Function Block Type cell (for example TOf), and press **F1**.

- Click **Check**  to check for errors.

Code Blocks

Both programming editors (programs and control modules) support code blocks. All the code blocks, except the first code block, are always self-defined which means that user defined code blocks can be created for structuring code. Thus, code blocks are a way of enhancing the readability, traceability, and structure, of the programming code. However, the number of code blocks must be kept to a minimum, else there is a risk of badly arranged programming structure.

The code blocks are always executed in a predetermined order, and in this example from left to right (programs).



Some characters are not allowed in Code block names. See the Online help for more information on the characters that must not be used in code block names.

Creating Code Blocks

1. Right-click the Code tab and select **Rename** from the context menu. A Rename Code Block window opens.

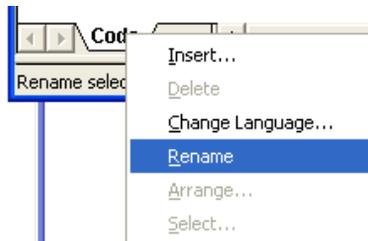
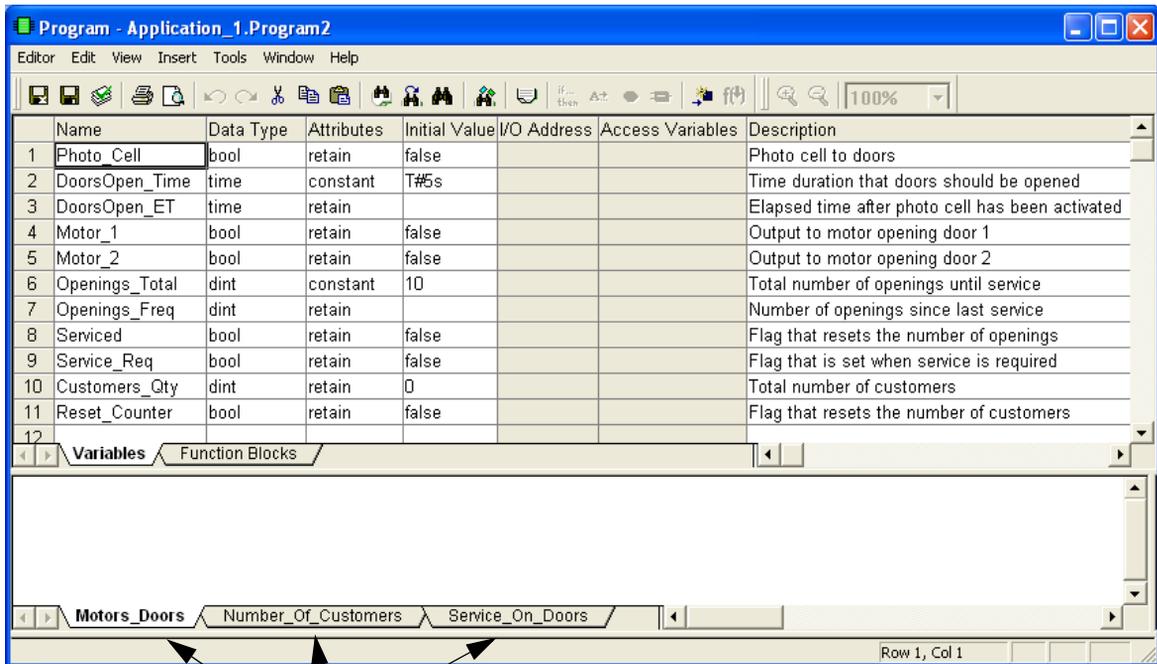


Figure 25. Right-click the code block tab to access the context menu

2. Write `Motors_Doors` in the Name field.
3. Click **OK**.
4. Right-click the `Motors_Doors` tab and select **Insert** from the context menu. A Insert New Code Block window opens.
5. Write `Number_Of_Customers` in the Name field and check that the Structured Text language is selected. Click **OK**.
6. Add a third code block in the same way and name it `Service_On_Doors`.



Code Blocks

Figure 26. The program editor with three new code blocks created

After creating the three code blocks representing motors, customers, and service issues in the program editor, the corresponding code block specific programming code can be written.



The code blocks are executed from left to right in the following sequence: Motors_Doors, Number_Of_Customers, Service_On_Doors.

Code Input

Making a Function Block Call in Motors_Doors

1. Select the code block tab **Motors_Doors** and place the cursor in the code pane.
2. Select **Insert > Variable, Type, Attribute** from the Menu bar (or press CTRL+J). A dialog list opens.



The variables and function blocks declared under the **Function Blocks** tab in the declaration pane are shown in the dialog list.

3. Select the **OpenDoors** function block in the list and press the ENTER key.



Type the initial letters of the Function block (for example, **Op** for OpenDoors), to choose the required function block in the scroll list.

4. Make sure the cursor is located directly after `OpenDoors` in the code pane. Enter a left-hand parenthesis ‘(‘.

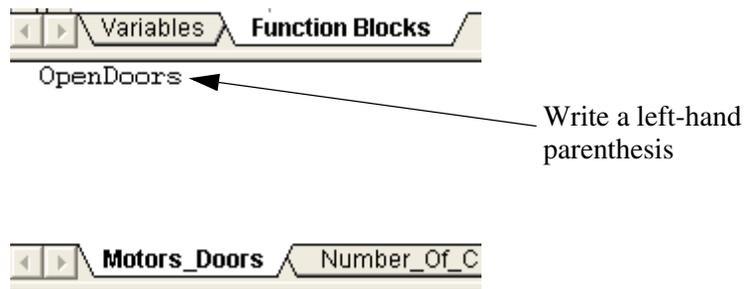


Figure 27. Write a left-hand parenthesis after `OpenDoors` in the code pane

- When the leading left-hand parenthesis ‘ (‘ is entered, a Function block call editor opens, see [Figure 28](#).

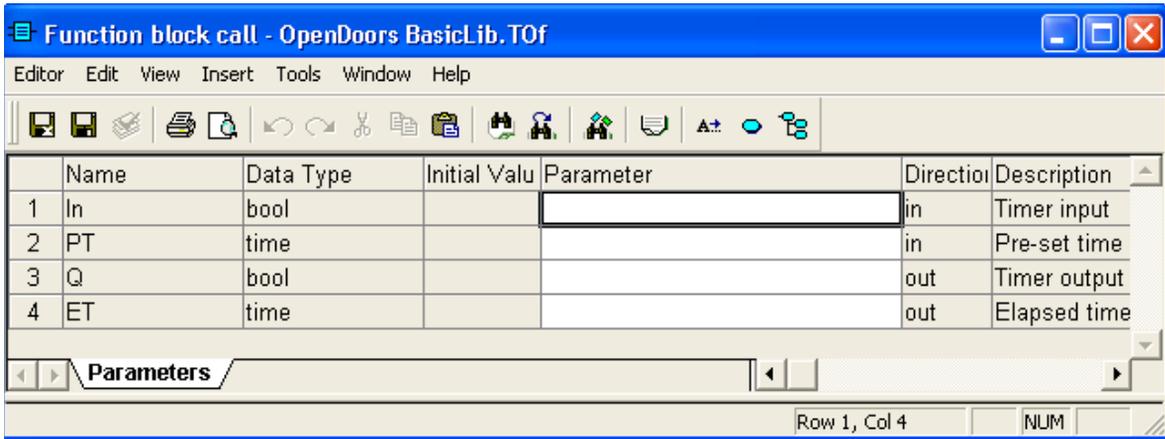


Figure 28. The Function block call editor

- Place the cursor in the first empty white cell and select **Insert > Parameter From List** (or press CTRL+J, or click  -icon in the Menu bar). A variable list opens.
- Select **Photo_Cell** and press the ENTER key to accept the selection.

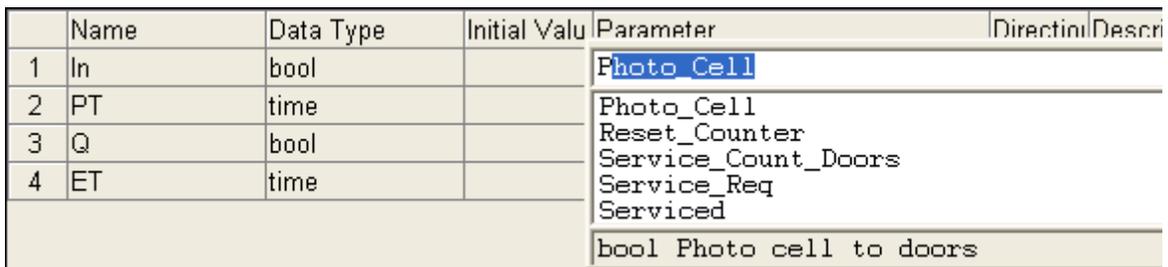


Figure 29. Variable list in the Function block call dialog

8. Fill in the other two variables in the parameter list according to [Figure 30](#).

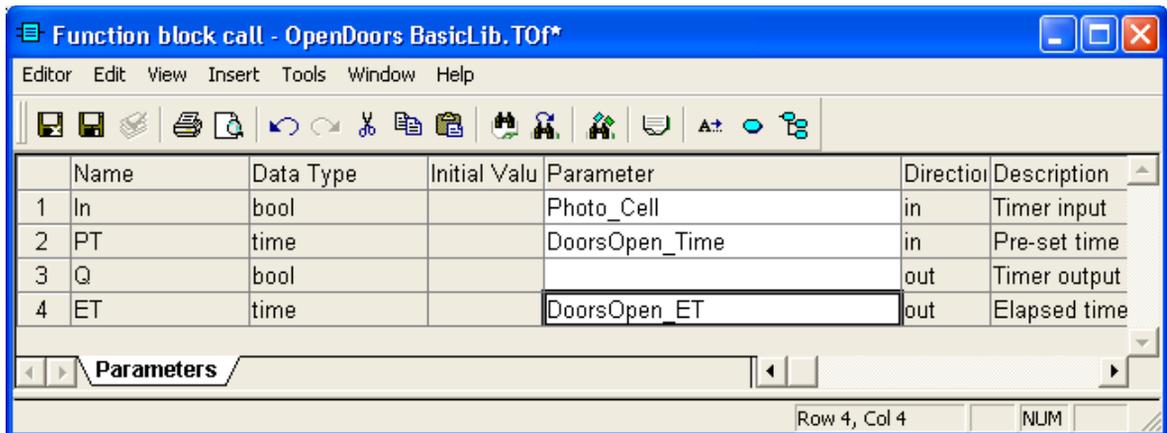


Figure 30. Connecting function block parameters

9. Click **Save and Close**  to insert the parameters into the code.

Q parameter in TOF

The output Q parameter is a Boolean signal, which represents the status on the door position (open or closed) and is passed on to the motors. For both doors to open, the Q signal must be passed to both motors. To achieve this, write the following code in the code pane:

```
Motor_1 := OpenDoors.Q;
Motor_2 := OpenDoors.Q;
```

The output Q is now addressed directly to the function block and a value assigned to both motors to open the doors, see [Figure 31](#).

```

OpenDoors( In := Photo_Cell,
           PT := DoorsOpen_Time,
           ET => DoorsOpen_ET );

Motor_1 := OpenDoors.Q;
Motor_2 := OpenDoors.Q;

```

Figure 31. The code block *Motors_Doors*



The code can be written structured with or without tabs and spaces.

10. Click **Save**  to save the code.

Making a Function Block Call in *Number_Of_Customers*

1. Select the code block tab *Number_Of_Customers* and place the cursor on the first line in the code pane.
2. Select **Insert > Variable, Type, Attribute** from the Menu bar (or press CTRL+J). A dialog list opens.



The variables and function blocks declared under the **Function Blocks** tab in the declaration pane are shown in the dialog list.

3. Select the *Customer_Count_Up* function block in the list.



Type the initial letters of the Function block (for example, **Cu** for *Customer_Count_Up*) to choose the required function block in the scroll list.

4. Accept the selection by pressing the ENTER key. Type the leading left-hand parenthesis '(' . A Function block call editor opens.
5. Place the cursor in the first empty white cell and select **Insert > Parameter From List** (or press CTRL+J). A variable list opens.

6. Select Photo_Cell in the list and press the ENTER key to accept the selection.

| | Name | Data Type | Initial Value | Parameter | Direction | Description |
|---|-------|-----------|---------------|--------------------------|-----------|-------------|
| 1 | CU | bool | | Photo_Cell | | |
| 2 | Reset | bool | | Photo_Cell | | |
| 3 | PV | dint | | Reset_Counter | | |
| 4 | Q | bool | | Service_Count_Doors | | |
| 5 | CV | dint | | Service_Req | | |
| | | | | Serviced | | |
| | | | | bool Photo cell to doors | | |

7. Fill in the other two variables in the parameter list according to Figure 32. Ignore the parameter for PV and Q.

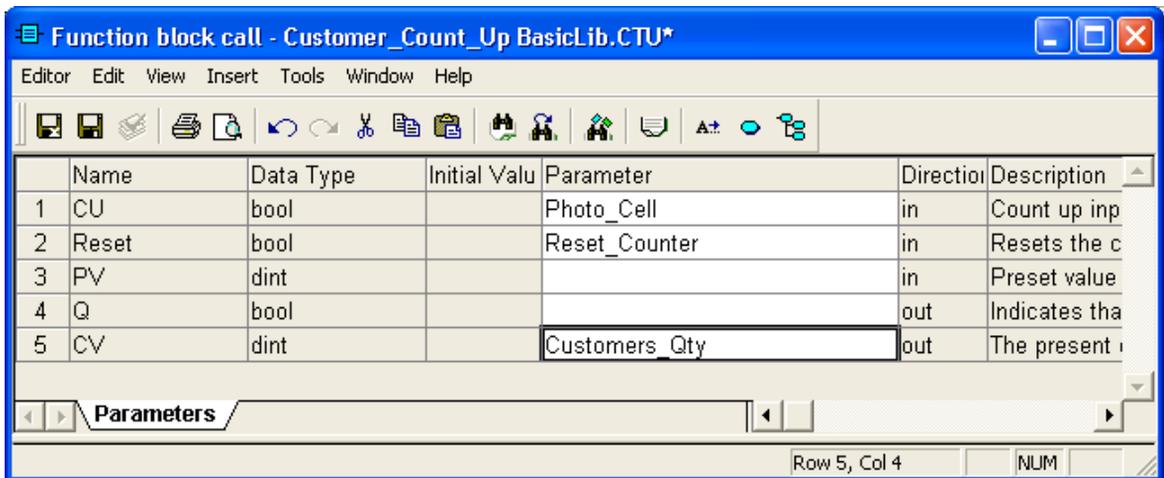
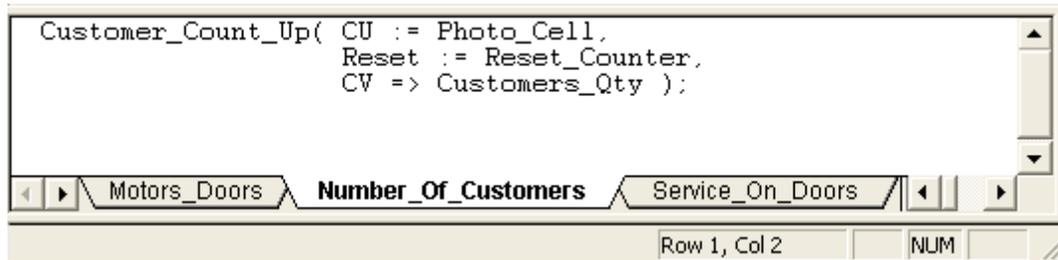


Figure 32. Connecting CTU function block parameters

8. Click **Save and Close**  to insert the parameters into the code. See Figure 33.



```
Customer_Count_Up( CU := Photo_Cell,  
                  Reset := Reset_Counter,  
                  CV => Customers_Qty );
```

The screenshot shows a code editor window with three tabs: 'Motors_Doors', 'Number_Of_Customers', and 'Service_On_Doors'. The 'Number_Of_Customers' tab is active, displaying the code block definition for the 'Customer_Count_Up' function. The status bar at the bottom indicates 'Row 1, Col 2' and 'NUM'.

Figure 33. The code block `Number_Of_Customers`

Making a Function Block Call in `Service_On_Doors`

1. Select the code block tab `Service_On_Doors` and place the cursor on the first line in the code pane.
2. Select **Insert > Variable, Type, Attribute** from the Menu bar (or press CTRL+J). A dialog list opens.



The variables and function blocks declared under the **Function Blocks** tab in the declaration pane are shown in the dialog list.

3. Select the `Service_Count_Doors` function block in the list.
4. Accept the selection by pressing the ENTER key. Enter the leading left-hand parenthesis '(', to open the parameter editor.

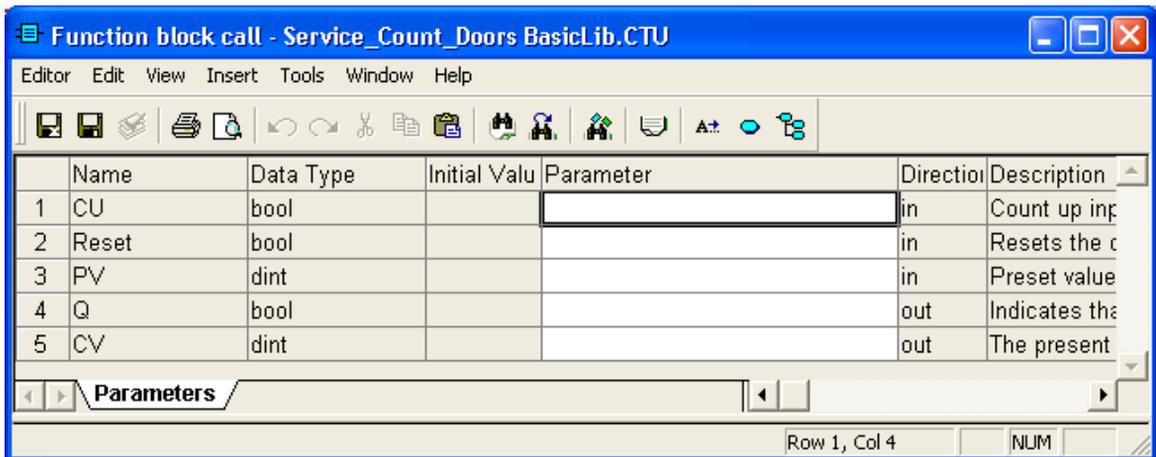


Figure 34. The Function Block Call Editor

- Place the cursor in the first empty white cell and select **Insert > Parameter From List** (or press CTRL+J). A variable list opens.
- Select `Motor_1` in the list and press the ENTER key to accept the selection.

| | Name | Data Type | Initial Value | Parameter | Direction | Description |
|---|-------|-----------|---------------|----------------|-----------|-------------|
| 1 | CU | bool | | Motor_1 | | |
| 2 | Reset | bool | | Motor_1 | | |
| 3 | PV | dint | | Motor_2 | | |
| 4 | Q | bool | | OpenDoors | | |
| 5 | CV | dint | | Openings_Freq | | |
| | | | | Openings_Total | | |
| | | | | bool | | |

- Fill in the other variables in the parameter list according to [Figure 35](#).

| | Name | Data Type | Initial Value | Parameter |
|---|-------|-----------|---------------|----------------|
| 1 | CU | bool | | Motor_1 |
| 2 | Reset | bool | | Serviced |
| 3 | PV | dint | | Openings_Total |
| 4 | Q | bool | | Service_Req |
| 5 | CV | dint | | Openings_Freq |

Figure 35. Connecting CTU function block parameters

8. Click **Save and Close**  to insert the parameters into the code. See [Figure 36](#).

```
Service_Count_Doors( CU := Motor_1,
                    Reset := Serviced,
                    PV := Openings_Total,
                    Q => Service_Req,
                    CV => Openings_Freq );
```

Navigation: Motors_Doors | Number_Of_Customers | **Service_On_Doors**

Figure 36. The parameters connected in the code block, *Service_On_Doors*



If an error message should be displayed in the message pane, double-click the error line to move the cursor to the error location in the code. Also, a brief description of the type of error is displayed in the message pane.

Testing MyDoors Project

Before downloading the application to a PLC and going online, it is necessary to first test the application in an offline mode to ensure that everything is working properly. This mode is called the Test Mode, where the Control Builder compiles and executes the code locally in the PC as if it is an AC 800M controller.

The test mode is an easy way to test the application many times. However, external communication is disabled during the test mode, thus reading and writing variables connected to IO units cannot be validated in test mode.



Before running the program in Test mode, a Difference Report window appears. However, the Difference Report function is not important for this example since it does not generate a report in Test mode. Choose to keep the default setting, or read how to turn off the function in the subsection [Disable/Enable Difference Report](#) on page 116. This example assumes that the Difference Report has the default setting enabled.

1. In Project Explorer, click **Test Mode** . The Test Mode Analysis window opens.
2. Click **Cold Restart All**.
3. Click **Continue**. A Difference Report window opens.

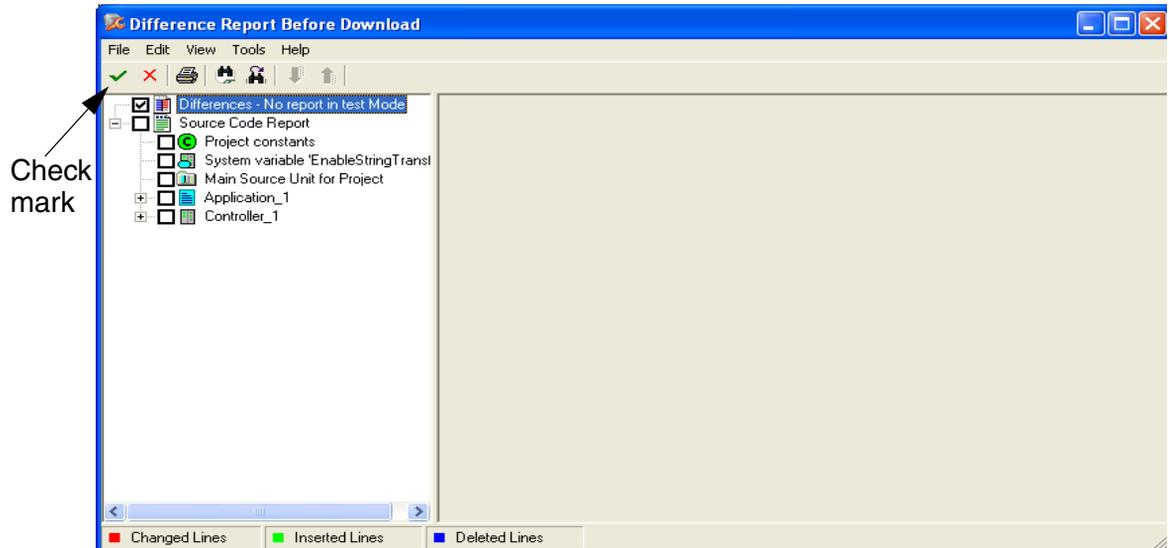


Figure 37. Difference Report Window

4. Select the green check mark (see [Figure 37](#)) to continue. Difference Report is not generated in Test mode.
5. Double-click **Program2** to display the editor.
6. Select **Motors_Doors** tab. All variables in **Program2** are listed in the upper pane and the code in the lower pane, see [Figure 38](#).

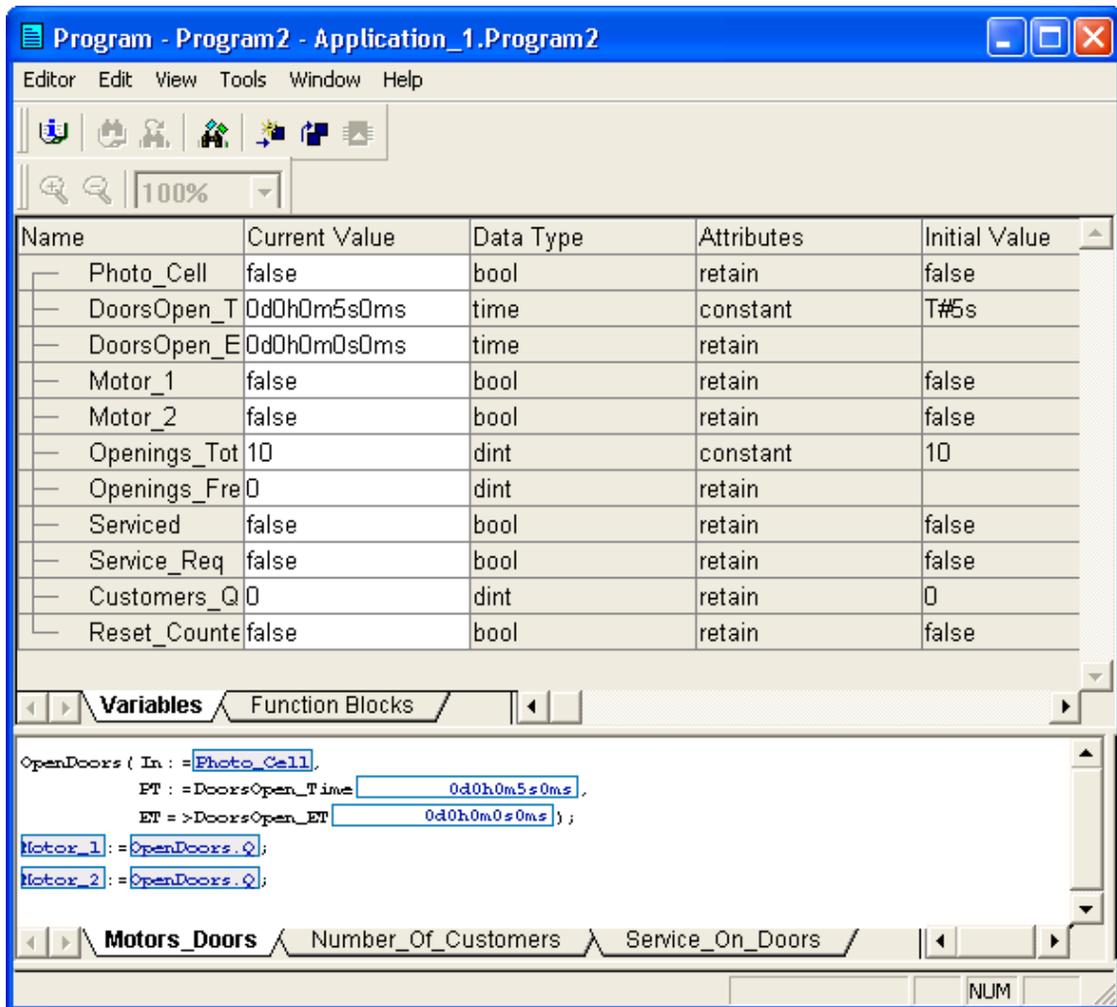


Figure 38. The Program2 editor in Test mode

Analyzing the Code During Program Executions

The test mode helps in testing and analyzing the project, without having a PLC ready in the Project Explorer tree. The variable values can be changed to study the program response.

Analyzing the variable conditions requires right-clicking of variables. These can be accessed either from the parameter list or from the code pane.

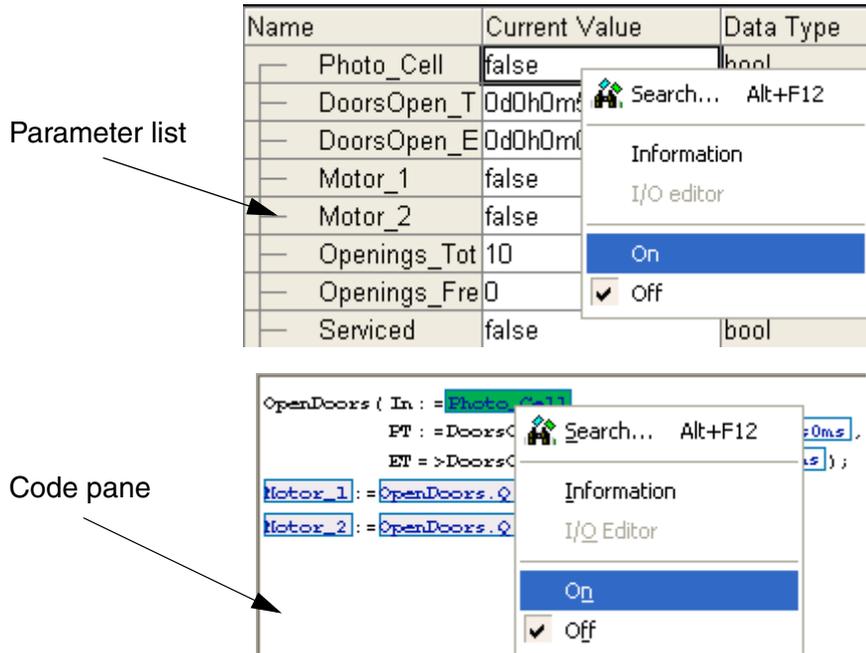


Figure 39. Changing the current value on a variable from the Parameter list, or in the code pane

1. Right-click `Photo_Cell` and select **On** in the context menu.

The motors change to True (start) and the number of openings since last service increases by one, as does the number of customers.

2. Right-click `Photo_Cell` and select **Off** in context menu.

Simulating that no customer is activating the photocell. The clock starts and counts up to five seconds at which point the motors are set to False (stop) and the doors close.

3. Right-click `Photo_Cell` and select **On**, then QUICKLY select **Off** again.

Simulating that a customer has activated the photocell. The number of openings is increased, as does the number of customers.

4. Wait until the doors close. Right-click `Photo_Cell` and QUICKLY select **On, Off, On, Off, On, Off**.

Simulating that three customers are passing the photocell one by one. Notice that the clock starts when the first customer passes the photocell and resets to 0 when the next customer passes. Consequently, the opening time is extended for a new period of 5 seconds, and so on. The number of times the doors open increases by one, whereas the number of customers increases by three.

This results in three openings of the doors and five customers.

5. In the variables list, right-click `Reset_Counter` and select **On**, then select **Off** again. Reset the customer counter as well.
6. Activate the photocell so that when the number of openings (`Openings_Freq`) passes, `Openings_Total.Service_Req` becomes *True*.
7. Right-click `Serviced` and select **On**, then select **Off** again.
Study the reaction of the counters and flags. The variable `Openings_Freq` resets.
8. Close Program editor.
9. From Control Builder Menu bar, select **Tools > Stop Test Mode**.

Project Examples in Control Builder

All the examples are Read-only, which means they cannot be altered. To study these, see [Opening the ShopDoors Example \(Read only\)](#) on page 67.

If these are used as a template or a skeleton for other projects, then the projects attribute needs to be changed from read only as described below.

Preparing the ShopDoors Example for a Project

From the disk where Compact Control Builder¹ is installed:

1. Start the Search tool in Windows and search for *ShopDoors_ST.prj*. The project file is located inside the example folder for the ShopDoors project.
2. Copy the complete **ShopDoors_ST** folder into the **Projects** folder².
3. Right-click the **ShopDoors_ST** folder and select **Properties** from the context menu.
4. Clear the folders properties Attribute Read-only. Apply changes to the folder, sub folder and files.
5. Follow the next steps under [Opening a Project](#) on page 66, to open the converted ShopDoors_ST project.

1. Compact Control Builder is normally installed on the Local Disk (C:)

2. The Projects folder is normally located at; C:\ABB Industrial IT Data\Engineer IT Data\Control Builder AC 800M\Projects

Opening a Project

1. From the Project Explorer, select **File > Open Project**. An Open Project window opens.

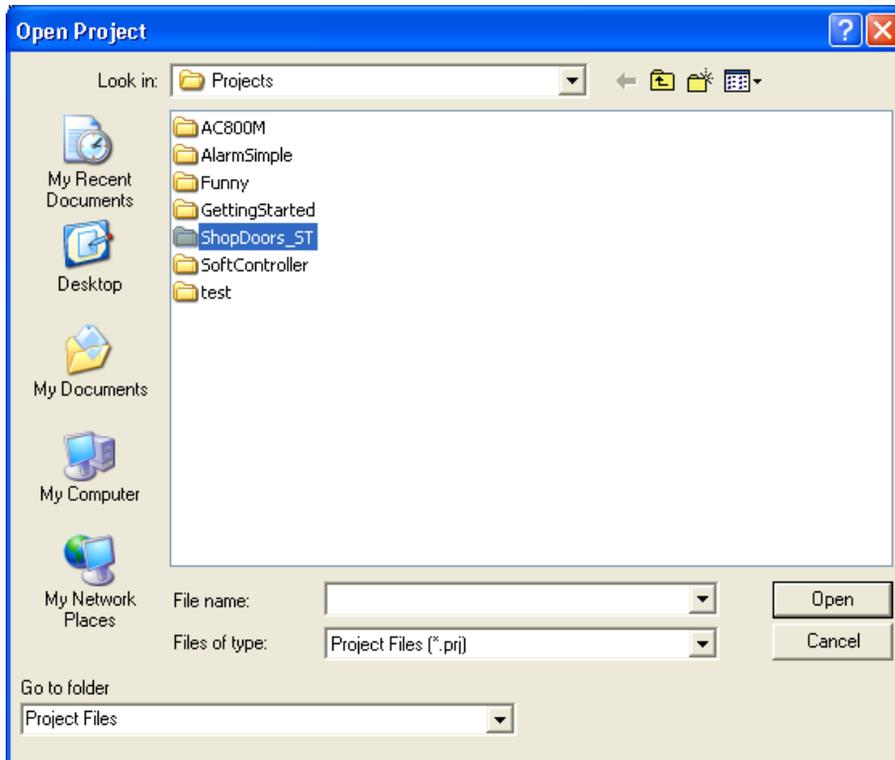


Figure 40. The Open project window

2. Select the **ShopDoors_ST** folder and click the **Open** button. The project ShopDoors_ST folder opens.
3. Select the **ShopDoors_ST.prj** file and click the **Open** button. The Project Explorer opens the ShopDoors_ST project.

Opening the ShopDoors Example (Read only)

1. From the Project Explorer, select **File > Open Project**. An Open Project window opens (Figure 40).
2. Press **Go to folder** drop-down menu and select **Example files**. The Example folder opens.

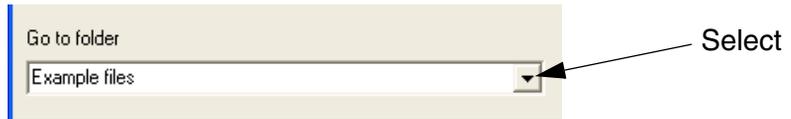


Figure 41. Drop-down menu for selecting project files or example files

3. Open the **ShopDoors_ST** folder and select the **ShopDoors_ST.prj** file.

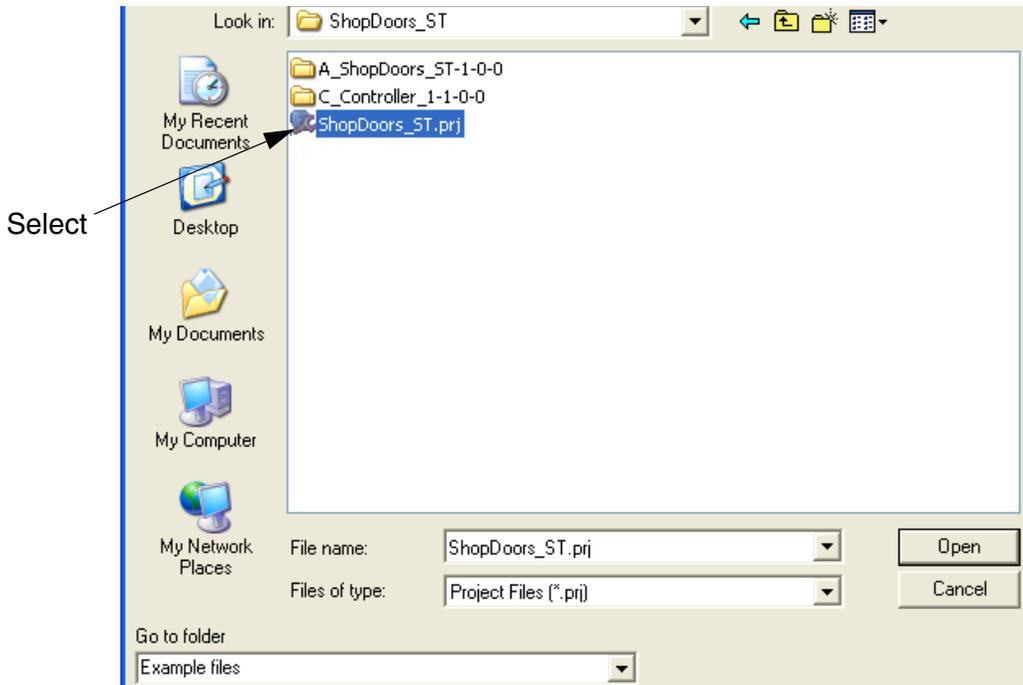


Figure 42. The Open project window, showing the contents of the ShopDoors_ST directory. Note the (ShoopDoors_ST.prj) project files

4. Click **Open**. The ready-made ShopDoors example opens in Project Explorer.

Section 5 Hardware Configuration

This section describes about adding or removing hardware units from the tree structure in the Project Explorer. It covers the necessary steps for building a software model that represents a limited part of a hardware configuration in the plant.

The Controller object in Project Explorer is the root to which hardware objects are added.

Configure Hardware

Study the hardware configuration in [Figure 43](#). Assume a PLC, together with six I/O modules. In this example, two of them are added to the tree structure in Project Explorer. Add DO814 and DI810, at positions 1 and 2 respectively.

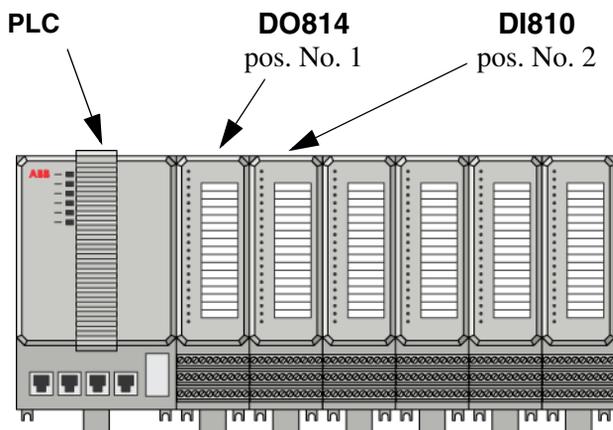


Figure 43. Hardware position for IO modules (for example DO814 at position 1 and DI810 at position 2)

Changing a CPU

The CPU connected in the Project Explorer must be same as in the controller, else the application cannot be downloaded to the Controller.

However, if the application is run with a SoftController, the choice of the CPU model is optional.



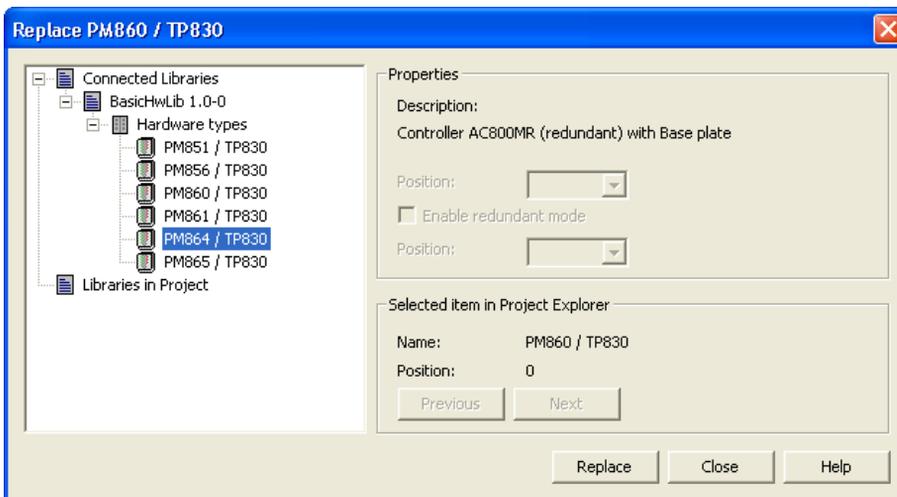
In this example, a default CPU PM860 is replaced with a CPU PM864.



To remove a hardware unit, right-click the object in the tree structure and select Delete.

To replace a CPU:

1. Expand **Controllers > PLC_1 > Hardware AC 800M** to view the **PM860 / TP830** item at position 0 in the Project Explorer tree.
2. Right-click the **PM860 / TP830** item and select **Replace Unit** in the context menu. A 'Replace' window opens.
3. Expand **Connected Libraries** and select, for example **PM864/TP830**.



4. Click **Replace** and then **Yes** to accept the change.

Adding the IO Modules DO814 and DI810

The S800 IO modules are represented in Control Builder as hardware types located in the hardware library **S800IOModulebusHwLib**. Thus, before adding the IO modules, first insert the hardware library to the project. Once the library is inserted to the project, connect the library to the hardware configuration and then access the IO modules and add them to the controller configuration.

To insert and connect a hardware library:

1. Expand **Libraries** folder to see the **Hardware** folder in the Project Explorer tree.

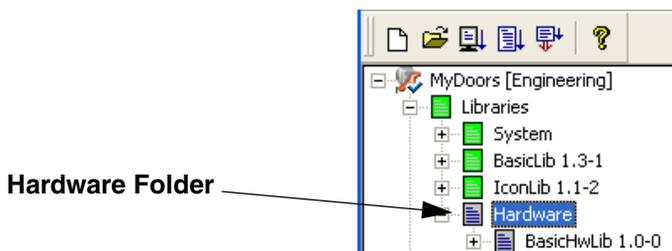


Figure 44. The hardware folder located inside Libraries folder in the Project Explorer

2. Right-click **Hardware** and select **Insert Hardware Libraries** from the context menu. A 'Insert Library' window opens.

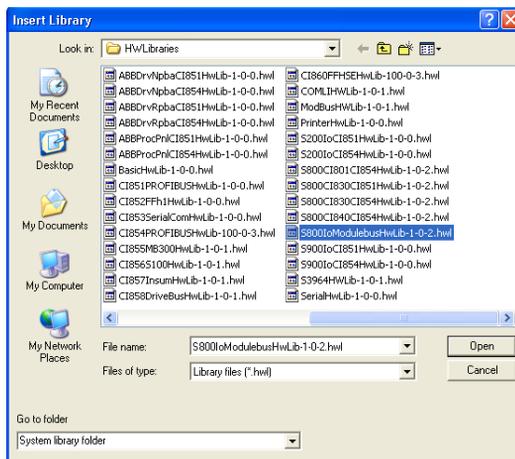


Figure 45. Selection of hardware libraries available in Compact Control Builder

3. Select the **S800IoModulebusHwLib** hardware library and click **Open** to insert a S800 Hardware Library to the project.



Figure 45 shows different hardware libraries for different configuration purposes. For example, the S800 IO library that is inserted contains S800 IO units for the Modulebus.

4. Expand **Controllers > PLC_1** to see the **Connected Hardware Libraries** folder in the Project Explorer tree.
5. Right-click the **Connected Hardware Libraries** folder and select **S800IoModulebusHwLib** from the drop-down menu.
6. Click **OK**.

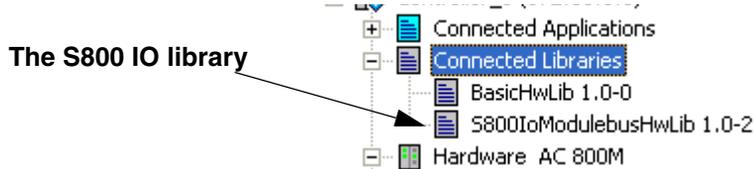
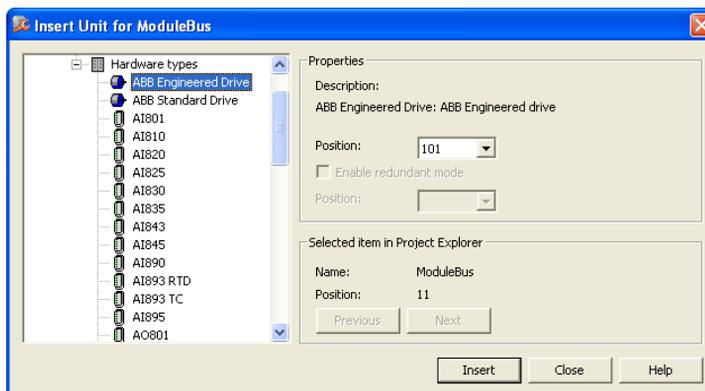


Figure 46. The new S800IO library has been connected to the controller

Adding the IO modules from the hardware library:

1. Expand **Controllers > PLC_1 > Hardware AC 800M > PM860/TP830** to see the **ModuleBus** item in the Project Explorer tree.
2. Right-click the **ModuleBus** item and select **Insert Unit** in the context menu. The 'Insert Unit for ModuleBus' window opens.
3. Expand **Connected Hardware Libraries > S800Io ModulebusHwLib > Hardware types** and select DO814.



4. Keep default position **1** from the *Position* drop-down menu and click **Insert**.
5. Select **DI810** from the list.
6. Keep default position **2** from the *Position* drop-down menu and click **Insert**.

When the two IO modules are added, the “hardware tree” looks like the configuration shown in [Figure 47](#).

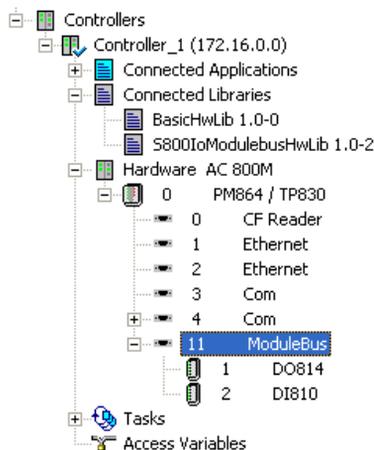


Figure 47. Hardware setup with the new IO Units added on the module bus



Information on an I/O unit, for example DO814 or DI810, can be accessible from the On-line Help; select the I/O unit in the Project Explorer and press F1.

Connect Variables to I/O Channels

Communication between I/O channels and code is established by connecting variables to I/O channels. Therefore, Control Builder provides two different connection methods. Both the methods can be followed for better understanding.



The MyDoors project is not completed unless both methods are followed.

[Method 1 - Using Dot Notation](#) on page 74 connects the variable Photo_Cell to the IO module DI810 by using dot notation. [Method 2 - Using a Path Selector](#) on page 75, connects the variables Motor_1 and Motor_2, to the IO module DO814 via a path selector menu.

Method 1 - Using Dot Notation

Connecting a Variable to DI810 Channel

From the Controllers in Project Explorer:

1. Double-click **DI810** I/O module. The DI810 hardware editor opens.
2. Select the **Connections** tab and place the cursor in the first empty white cell.
3. Type **A** (for Application_1) and observe how the editor fills in the rest.
4. Press “.” (dot) to move to the next level. All three Programs are displayed.
5. Select **Program2**, and press “.” (dot) again. A list of variables opens.
6. Select Photo_Cell in the list.
7. Press the ENTER key. The Photo_Cell variable has been connected to the first channel in DI810.



Figure 48. The variable `Photo_Cell` connected to first IO Channel

Method 2 - Using a Path Selector

Connecting a Variable to DO814 Channel

From the Controllers in Project Explorer:

1. Double-click the **DO814** I/O module. The hardware editor for DO814 opens.
2. Select the **Connections** tab and place the cursor in the first empty white cell (Channel QX0.11.3.1 in the **Variable** column).
3. Right-click **Insert** > **Insert Path From Tree** from the context menu. A list box opens.



The Menu bar action **Insert** > **Insert Path From Tree**, can also be done by simply clicking the  icon located in the hardware editor.

4. Expand **Application_1** > **Program2**. Double-click **Motor_1** to insert the full path, see [Figure 49](#).

| Channel | Name | Type | Variable |
|-------------|----------------|-------|----------|
| QXD.11.3.1 | Output 1 | bool | |
| QXD.11.3.2 | Output 2 | bool | |
| QXD.11.3.3 | Output 3 | bool | |
| QXD.11.3.4 | Output 4 | bool | |
| QXD.11.3.5 | Output 5 | bool | |
| QXD.11.3.6 | Output 6 | bool | |
| QXD.11.3.7 | Output 7 | bool | |
| QXD.11.3.8 | Output 8 | bool | |
| QXD.11.3.9 | Output 9 | bool | |
| QXD.11.3.10 | Output 10 | bool | |
| QXD.11.3.11 | Output 11 | bool | |
| QXD.11.3.12 | Output 12 | bool | |
| QXD.11.3.13 | Output 13 | bool | |
| QXD.11.3.14 | Output 14 | bool | |
| QXD.11.3.15 | Output 15 | bool | |
| QXD.11.3.16 | Output 16 | bool | |
| QWD.11.3.17 | All Outputs | dword | |
| IWD.11.3.18 | Channel status | dword | |
| IWD.11.3.19 | UnitStatus | dint | |
| | | | bool |

Figure 49. The path for Motor_1 variable selected in the tree

- Place the cursor in the second empty white cell (Channel QXD.11.3.2 in the **Variable** column) and connect Motor_2. Use the  icon.

After connecting the variables, the hardware editor looks as in [Figure 50](#).

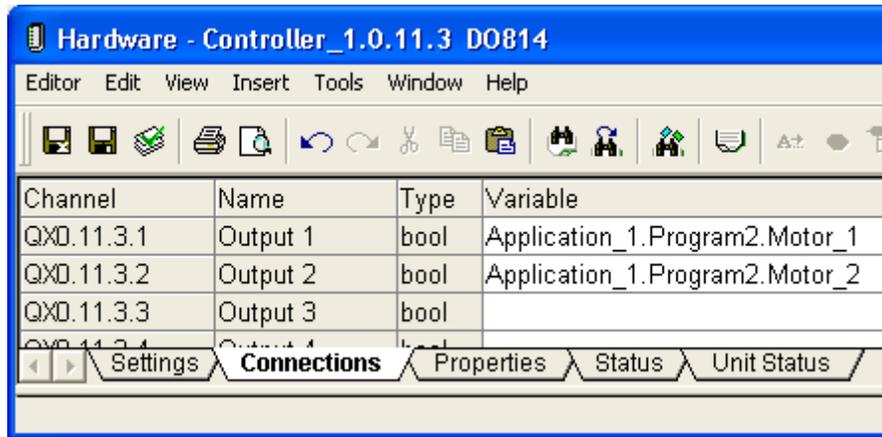


Figure 50. The Motor_1 and Motor_2 connected to D0814

6. Click **Check**  to check the declaration.
7. Click **Save and Close** .

Reading I/O addresses from the Application

An easy way to read the I/O address is to open **Program2** in the program editor and check the column labeled **I/O Address**. This column shows the address for the photocell and the motors, as shown in [Figure 51](#).

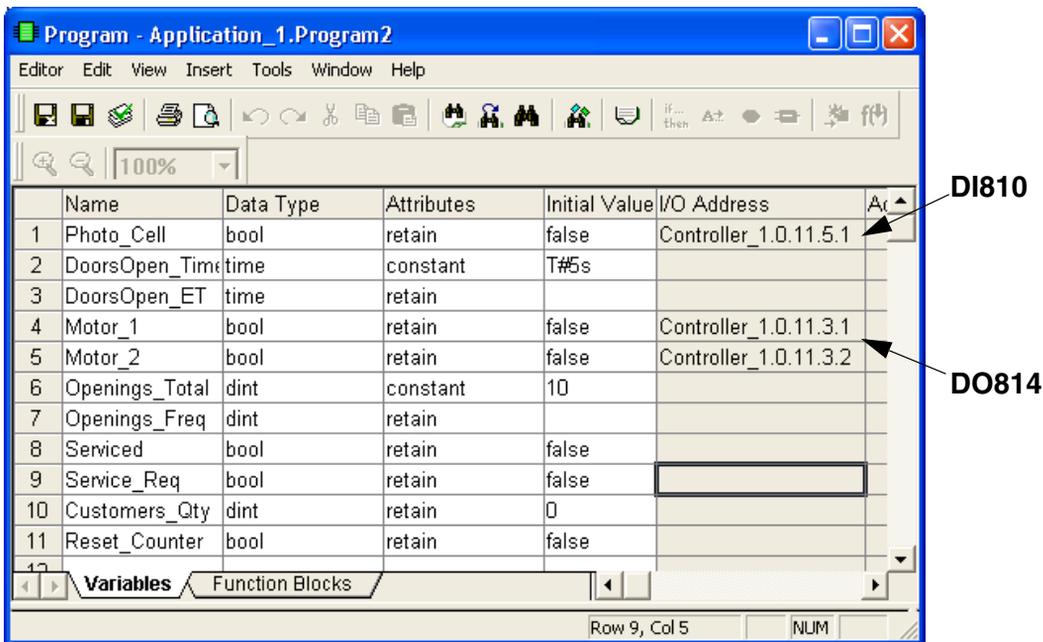


Figure 51. The I/O Address column shows how variables are connected to I/O channels

Changes made to I/O connections in the hardware editor are reflected in both editors.

The project has now been tested offline and the hardware configuration is complete.

Section 6 Connecting the PLC and Go Online

This section describes the prerequisites for connecting a PLC and the general procedure for downloading a project to the PLC.

Before completing the MyDoors project by downloading the application to a PLC, create a project according to [Section 4, MyDoors Project](#). Then, follow the instructions in [Section 5, Hardware Configuration](#).

If there is no access to a PLC or IO modules, use a **SoftController**. In that case, refer to [Setting the System Identity in Control Builder](#) on page 86.

Firmware Upgrade

The PLC firmware version and the Compact Control Builder version must be identical. If the firmware version is different, perform the steps in this section to upgrade the firmware.



Firmware upgrade can be performed from the Compact Control Builder via the Ethernet network (see the Control Builder Online Help). The Serial line upgrade procedure is also available.

Firmware Upgrade via the Serial Cable (TK212A)

1. Connect the serial cable between the Control Builder PC and the PLC, as specified in [Table 2](#). For the type of cable, see [Appendix D, Communication Cables](#).

Table 2. Cable connection for the PLC

| PLC | Tool Port | Connector | Cable Name |
|---------|-----------|-----------|------------|
| AC 800M | COM 4 | RJ 45 | TK212A |



During the upgrade, do not run any program capable of blocking the selected COM port. This applies in particular to the MMS Server program.

2. Turn on the power to the PLC.
3. From the Windows Start menu select **All Programs > ABB Industrial IT AC 800M > Utilities > Serial Firmware Upgrade**. The following dialog box appears.

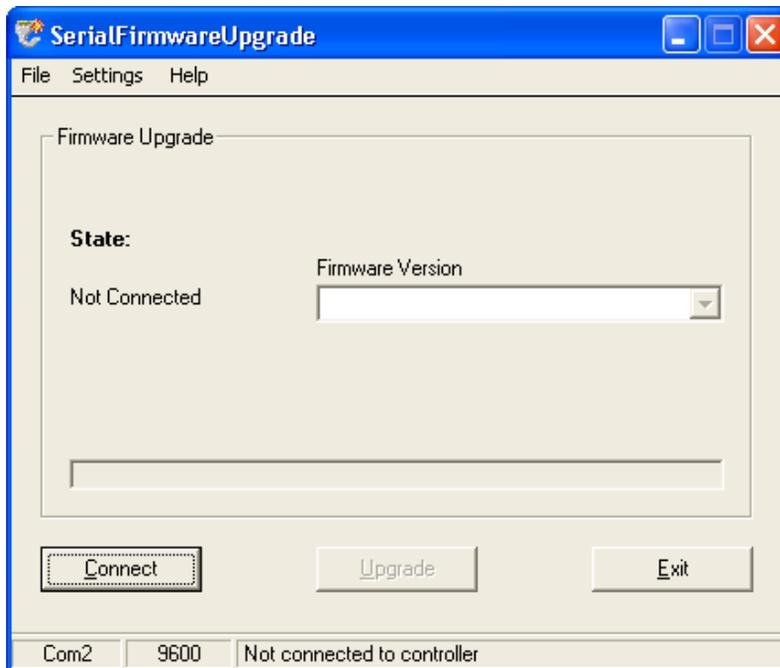


Figure 52. The Serial Firmware Upgrade dialog box

4. Select **Settings > COM Port** from the drop-down menu. Make sure the settings correspond to the physical COM port (on the PC) to which the cable is connected.

- Click **Connect** button and press the **Init** push-button on the PLC until the **Run** LED starts to blink. Wait for a minute until a message appears. If the connection is successful, a confirmation text appears in the Firmware Version text field (Figure 53).

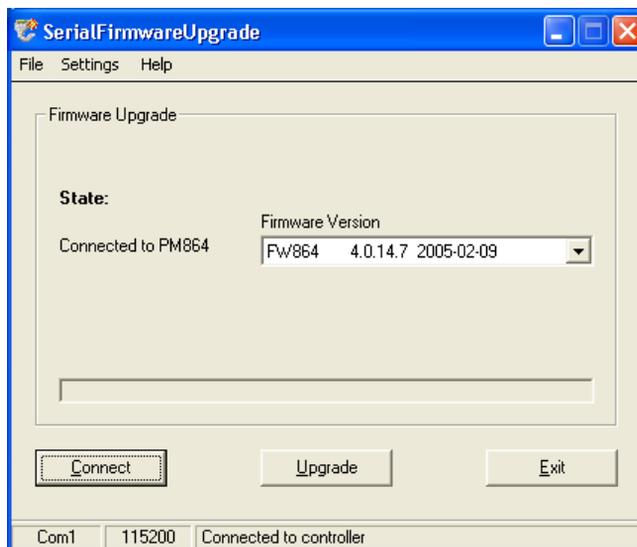


Figure 53. A Firmware version displayed in the text field



If an error message “Connection failed” appears, check the cables and repeat the steps again.

- Select Firmware version¹ from the drop-down menu and click **Upgrade**. File transmission starts to the PLC. A confirmation window opens when the PLC is upgraded, see Figure 54.

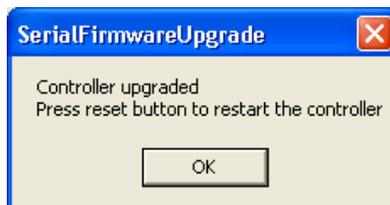


Figure 54. The Serial Firmware Upgrade window

1. The firmware version must be identical to the installed Control Builder version.

7. Click **OK**.
8. Click **Exit**.
9. Press the **Init** push-button on the PLC until the **Run** LED starts to blink.

Setting an IP Address

A unique PLC IP address must be set to avoid conflict with other devices on the Control Network. This subsection describes setting up an IP address for the PLC using the serial cable (TK212A), without any connection to the network. Furthermore, it provides instructions to setup the PC that runs the Control Builder. However, these instructions are strictly Microsoft Windows specific. A configuring tool, named IPConfig, is used to set the IP address for the PLC.

Setting IP Address for PLC

Preparations

Connecting the cable between the Control Builder and the PLC are exactly the same as described in [Firmware Upgrade](#) on page 79.

1. Connect a serial cable between the Control Builder PC and the PLC, as specified in [Table 3](#). For the type of cable, see [Appendix D, Communication Cables](#).

Table 3. Cable connection for the PLC

| PLC | Tool Port | Connector | Cable Name |
|---------|-----------|-----------|------------|
| AC 800M | COM 4 | RJ 45 | TK212A |



During the upgrade, any program that is capable of blocking the selected COM port should not be run. This applies in particular to the MMS Server program.

2. Turn on the power to PLC.

Starting the IPConfig Tool

1. From the Windows Start menu select **All Programs > ABB Industrial IT AC 800M > Utilities > IPConfig**. An IP Config dialog opens.

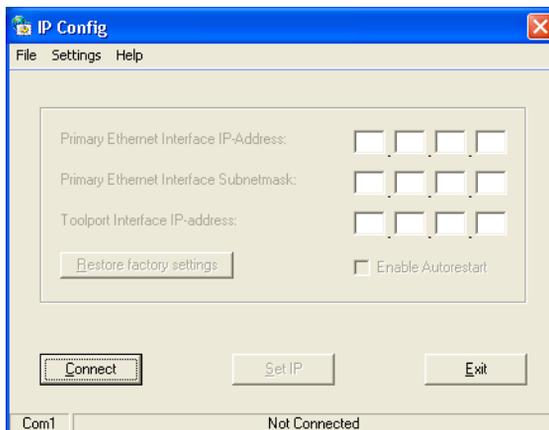


Figure 55. The IP Config dialog box

2. Click the **Connect** button and press the **Init** push-button on the PLC until the **Run LED** starts to blink. Wait for a minute until a message appears, see [Figure 56](#).



Figure 56. The IPConfig dialog box with factory default setting



If the error message “Connection failed” appears, check the cables and repeat the steps again.

- From the IP Config dialog menu, select **Settings > Advanced Mode**.
- Enter a unique IP address (obtainable from the Control Network Administrator). Example: 172.16.84.124, see [Figure 57](#).

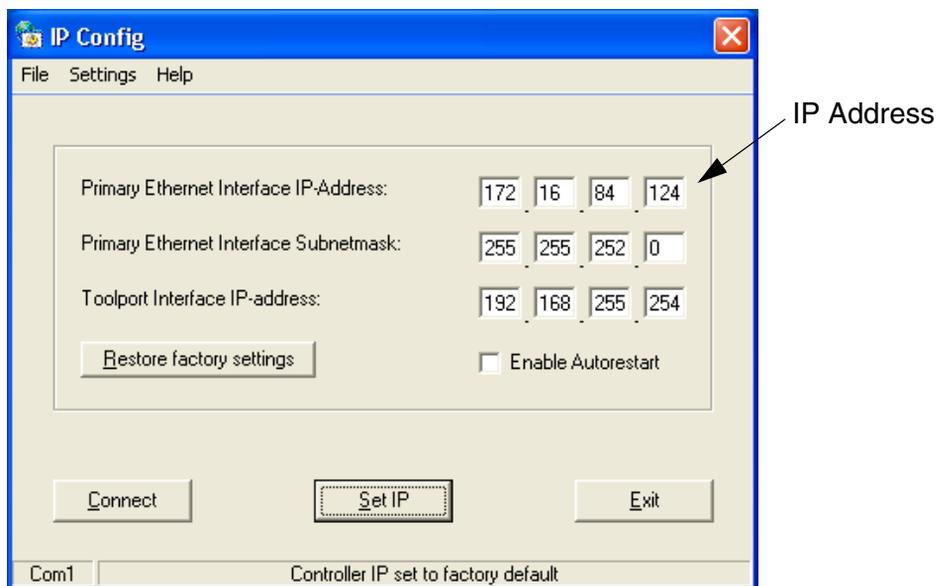


Figure 57. IP Config window for setting unique IP address

- Type Primary Ethernet Interface Subnetmask (255.255.252.0) and click **Set IP**. The new address is sent to the PLC and an IP Config window opens, see [Figure 58](#).

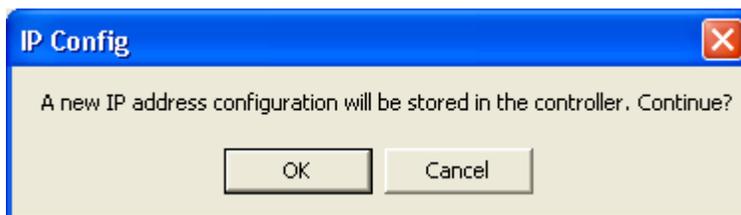


Figure 58. IP Config confirmation dialog window

- Click **OK**.
- Press the **Init** push-button on the PLC until the **Run** LED starts to blink. The new IP address is not valid until the PLC is restarted.

Setting IP Address for PC

The following instructions help in setting up the IP address (in Windows) for the Control Builder PC.

1. Select **Start > Settings > Control Panel**.
2. From Control Panel in Category View, select **Network and Internet connections** and then select **Network Connections**.



The Control Builder uses the IP address of the last network adaptor. Hence, setup the correct order of network adaptors in the Control Builder PC, so that Control Network adaptor comes last.

3. Right-click **Local Area Connection** and select **Properties**. A Local Area Connection Properties dialog opens.
4. In the 'Connect using' list, select the Ethernet board.
5. Select **Internet Protocol (TCP/IP)** and click **Properties**. An 'Internet Protocol (TCP/IP) Properties' dialog opens.
6. Select **Use the following IP address**.



The PC and PLC NetID must be the same for the first three positions (start from left to right). For example, if the PLC has the IP address 172.16.84.124, then the PC must have the IP address 172.16.84.Q. The number represented by Q must not be same as in the PLC (124 in this example).

7. Enter an IP address, in this example (172.16.84.120) and then enter Subnet mask (255.255.252.0).
8. Click **OK** to close all dialog windows.
9. Connect a network cable. The port and channel positions are shown in [Table 4](#).



To check that the IP configuration works; open the command prompt DOS window and ping the PLC by writing the following command: ping 172.16.84.n, where "n" is the address selected for the PLC.



If the PLC is to be connected to a PC via a switch or hub, then a *straight-through* Ethernet cable should be used. If there is a direct connection between the PLC and the PC, then use a *cross-over* Ethernet cable.

Table 4. Channel positions for connecting the Ethernet cable in the PLC

| PLC | Communication Interface | Position | Channel |
|---------|-------------------------|----------|---------|
| AC 800M | Built-in | - | CN1 |



CN2 port on the PLC must not be connected to the network. This port is used for connecting the PLC to a secondary network.

Downloading the Project via Ethernet

The connection with the PLC is established, once the upgraded firmware (Serial Firmware Upgrade) is downloaded and the IP addresses are given. This means that the application is ready to download projects to the PLC and Go Online.

Setting the System Identity in Control Builder

To download projects to the PLC, first set the system identity in Project Explorer.

Setting the IP address for PLC_1

1. In the Project Explorer, expand **Controllers**.

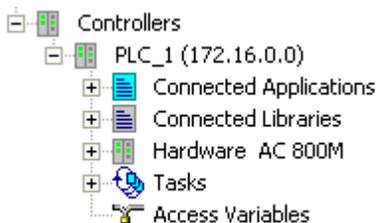


Figure 59. The Controllers expanded in Project Explorer

2. Right-click **PLC_1** and select **Properties > System Identity** from the context menu. The System Identity window opens.

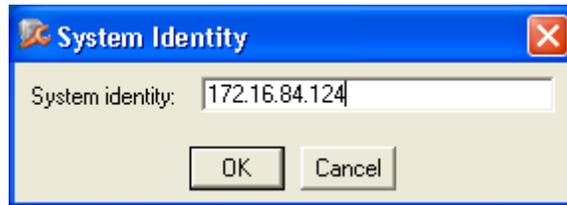


Figure 60. The System Identity window for setting the IP address

3. Enter the IP address of the PLC (for example: 172 . 16 . 84 . 124) and click **OK**. The System Identity window closes.



If the application is run with a **SoftController**, then enter the computer's IP address and add a colon and the digit 2. Example: 10.46.35.117:2

4. Expand **Hardware AC 800M** to view **1 Ethernet**. Right-click the **Ethernet** icon (at position 1) and select **Editor** to open the editor.
5. Select the **Settings** tab (lower left corner, see [Figure 61](#)) and enter the IP address in the IP address Value field.



Note that the IP address of the first Ethernet port has to be the same as the IP address of the PLC (system identity). The second Ethernet port (at position 2) is only used if the PLC is connected to a redundant network. For more information about redundant networks, study the subsection [Setting Up Redundant Network](#) on page 119

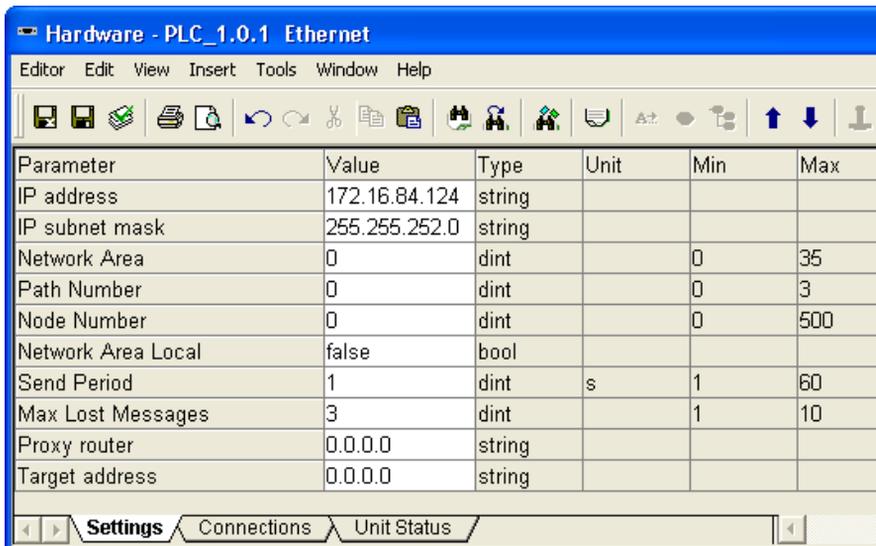


Figure 61. IP address for PLC Ethernet port at position 1 (in this case 172.16.84.124 which is the same IP address as given in System Identity)

Click **Save and Close** .

Downloading the Project to a PLC

Ensure that the project has no errors, before downloading the application to a PLC.

If the application is run with a SoftController, see [Downloading to a SoftController](#) on page 89.



Ensure that the PLC and all other hardware units have the right firmware. For instructions on how to check and upgrade firmware versions, see [Firmware Upgrade](#) on page 79.

Downloading to the PLC

The following instructions address the project MyDoors, which was previously created in [Section 4, MyDoors Project](#). However, these instructions are common for downloading any project application.

1. Ensure that the MyDoors project is in Offline mode.

2. Click **Download Project and Go Online** . The Online analysis window opens.
3. Click **Cold Restart All**.
4. Click **Continue**. A Difference Report dialog opens (unless it has been disabled).

The Difference Report function is enabled by default, thus a window opens unless disabled.

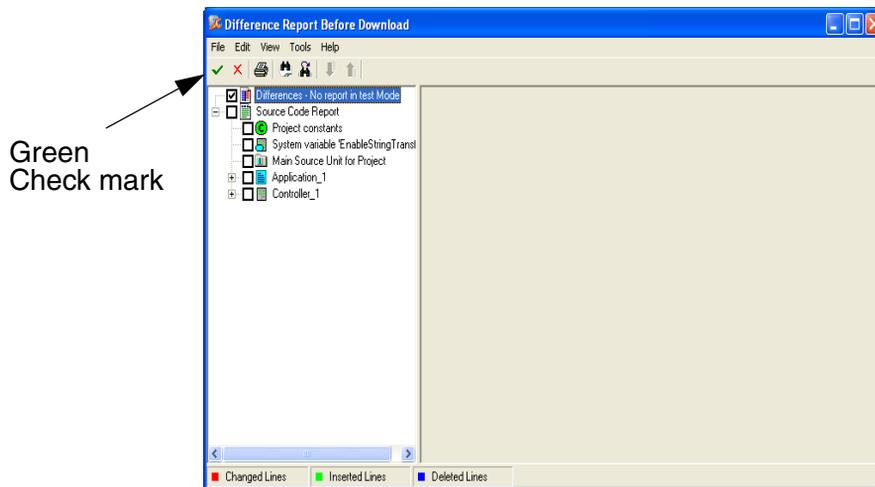


Figure 62. Difference Report window

5. Click the green check mark (see [Figure 62](#)) to continue.

Downloading to a SoftController

Make sure that the project (in this example, MyDoors) is in Offline mode (not running in test mode). From Project Explorer, expand the Controllers folder:

1. Right-click **PLC_1** and select **Simulate Hardware** from context menu.
2. Start the SoftController. Double-click the SoftController icon on the desktop (if desktop shortcut is selected during installation), or from the Start menu on the Windows Task Bar: **Start > All Programs > ABB Industrial IT AC 800M > SoftController > SoftController**. The SoftController start panel opens.

3. Click the **Start** button. The Status field displays *Started* and the SoftController starts.

From Project Explorer:

4. Click **Download Project and Go Online** . The Online analysis window opens.
5. Click **Cold Restart All**.
6. Click **Continue**.

Test the Program Online

This subsection describes how to force the variable Photo cell that is connected to the IO unit DI810 in the MyDoors sample project.

The *forcing* function can be used to activate/deactivate an I/O.

1. In Project Explorer, right-click **Program2** and select **Online Editor** to open the online editor.
2. Right-click I/O module **DI810** and select **Editor**.
3. In the Status tab, check the box in the **Forced** column, see [Figure 63](#). Change the variable `Photo_Cell` value to 1 (true), return quickly to 0 (false) and inspect the motor's values in the online editor (values change to 1 for five seconds and then return to 0).

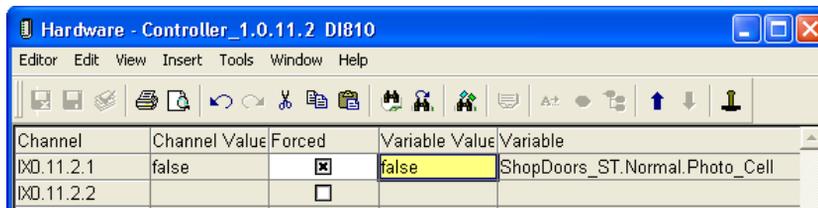


Figure 63. The status of the photocell and the motors can be forced in the I/O editor

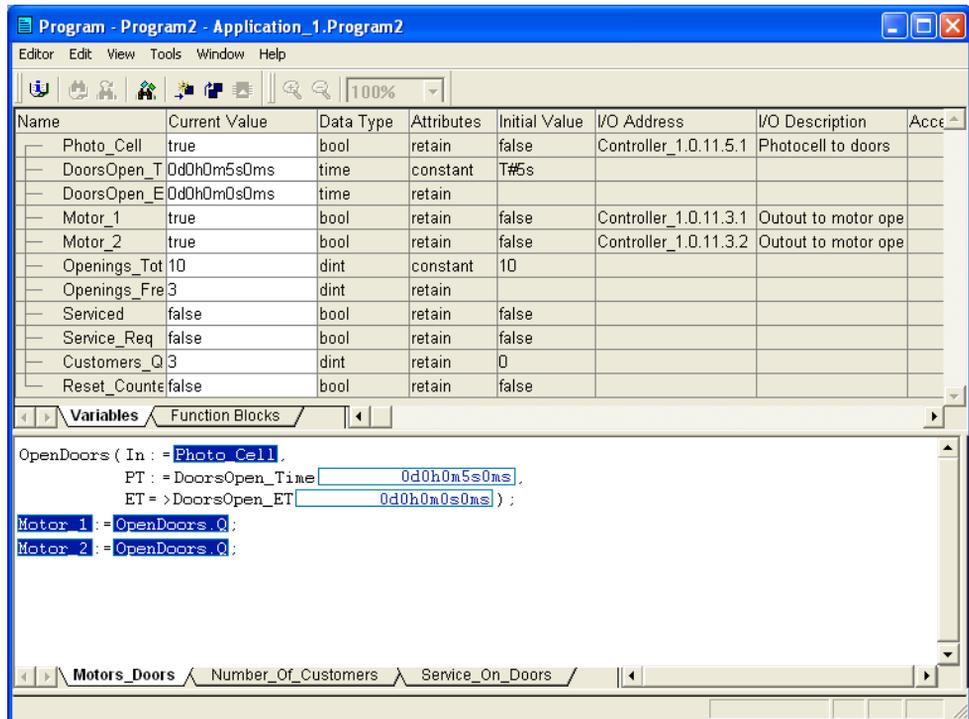


Figure 64. The changing motor values can be inspected in the online editor

The MyDoors project is now completed.

Appendix A Compact Control Builder AC 800M Settings

This appendix contains information about the Setup Wizard for Compact Control Builder. If a single-user configuration is set up with minor adjustments, then the wizard guides the user through the settings. However, if a multi-user configuration is to be setup, then the file path in the last dialog, 'File Location' (under Product settings), needs to be changed.

This appendix contains:

- [Starting the Setup Wizard for Compact Control Builder](#) on page 94.
- [Product Settings for Compact Control Builder](#) on page 94.
- [Multi-User Configuration](#) on page 98.
- [Download](#) on page 111.
- [Disable/Enable Difference Report](#) on page 116.



If the default settings for a single-user configuration are to be used, there is no need to configure the settings in the two Setup Wizards in this appendix.



Exclude the ABB Industrial IT Data folder and any used shared network disk from a virus scan, if the files are scanned at access. The user must configure the anti-virus program to scan these files and folders on demand or at a scheduled scan.

Starting the Setup Wizard for Compact Control Builder

From **Start** select **All Programs > ABB Industrial IT AC 800M > Utilities > Setup Wizard**.



Some tabs have an Apply button. Specified settings are not implemented until this button is clicked.

All *Setup Wizard* dialog boxes contain a Show Settings button. Click this button to open a log file on screen containing all available Wizard settings. It also contains a list of system environment variables.

Product Settings for Compact Control Builder

Memory Reservation

The total memory reserved for an application is based on a calculation involving the size of the physical RAM memory and the hard disk paging file. The following is shown in the Memory topic (see [Figure 65](#)).

- Physical RAM memory.
- Total paging file (this file can be changed in the Windows control panel).
- Maximum recommended heap.
- Actual heap (should normally be increased).

The actual heap must be set to the size of the application, but it must not exceed the Maximum recommended heap size, see memory settings in [Figure 65](#).

The default Compact Control Builder heap size is 256 MB. However, in conjunction with large projects, this size is not sufficient and it results in failure of the Compact Control Builder. If this failure occurs (if the Control Builder does not start), increase the heap size to double the value of the previously allocated heap size.



The amount of free memory can be checked by opening the About... dialog box in the Control Builder Help menu. Heap size should be increased when less than 30% remains.

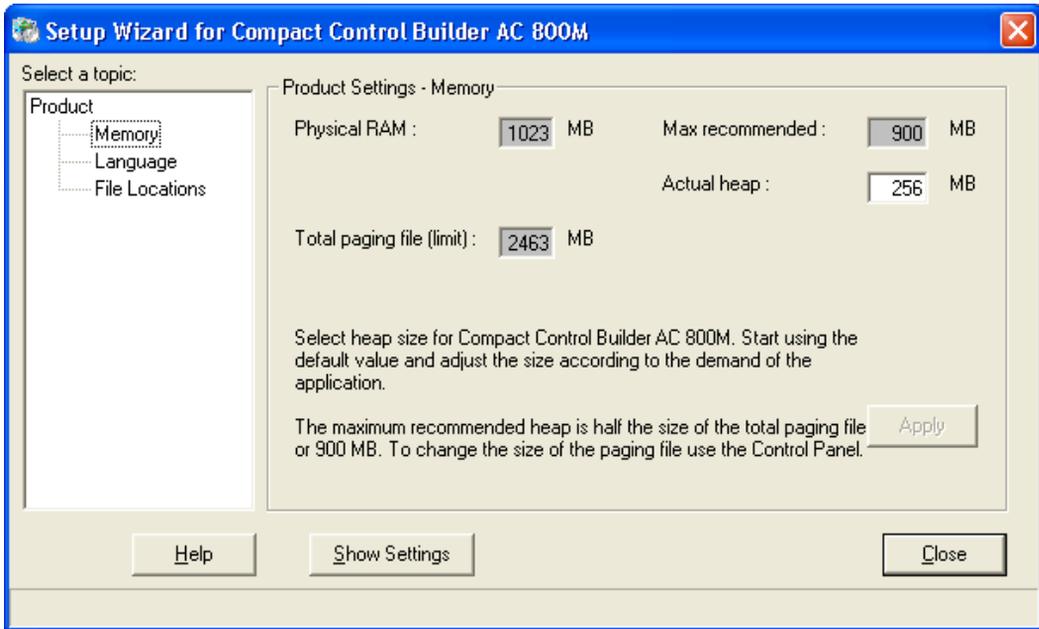


Figure 65. Setting the memory Heap size dialog. Click the topic 'Language' to proceed to next step in the wizard.



Ensure to click **Apply** to activate the changes.



A heap size must not be larger than the paging file. When the heap size is saved, the system checks this.

If the heap size is larger than the maximum recommended (half the total paging file or 900MB) size, a warning message appears to change the size. If the heap size is larger than the Total paging file, an error message appears to reduce the size.

Language

The language to be used for the programming tool, the libraries, and the Online Help files is selected under the Language tab (see [Figure 66](#)). By default, the installation program selects English.

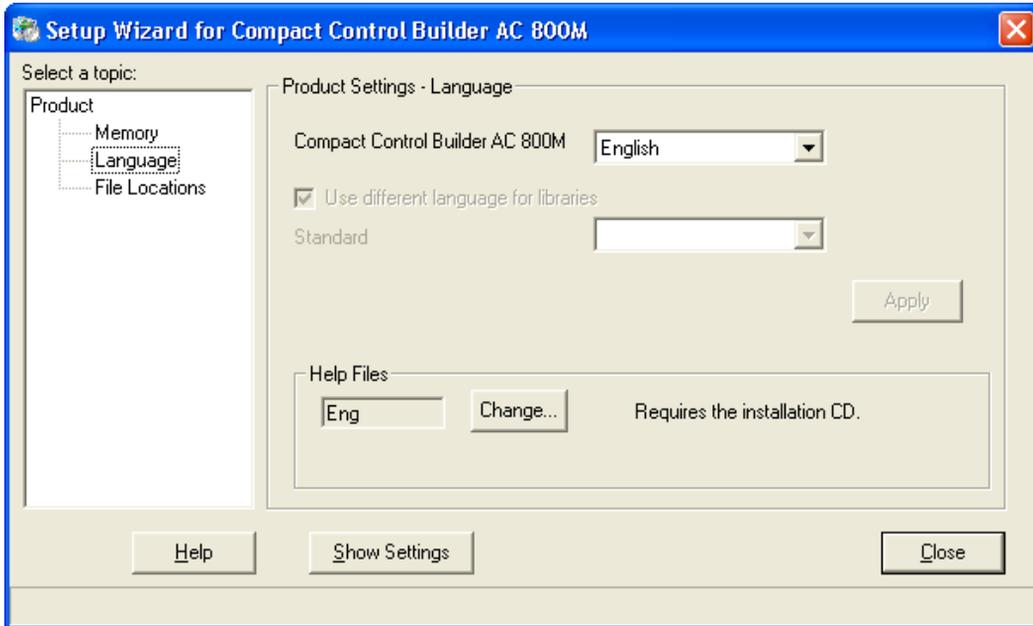


Figure 66. The Product setting dialog for language. Click the topic 'File Locations' to proceed to next step in the wizard.

File Locations

Compact Control Builder has three file locations which can be managed by the Setup Wizard. The Working folder and the MMS server working folder are handled by the system, thus should never be modified. The Project folder should only be changed, if the Compact Control Builder station should be part of a multi-user configuration.

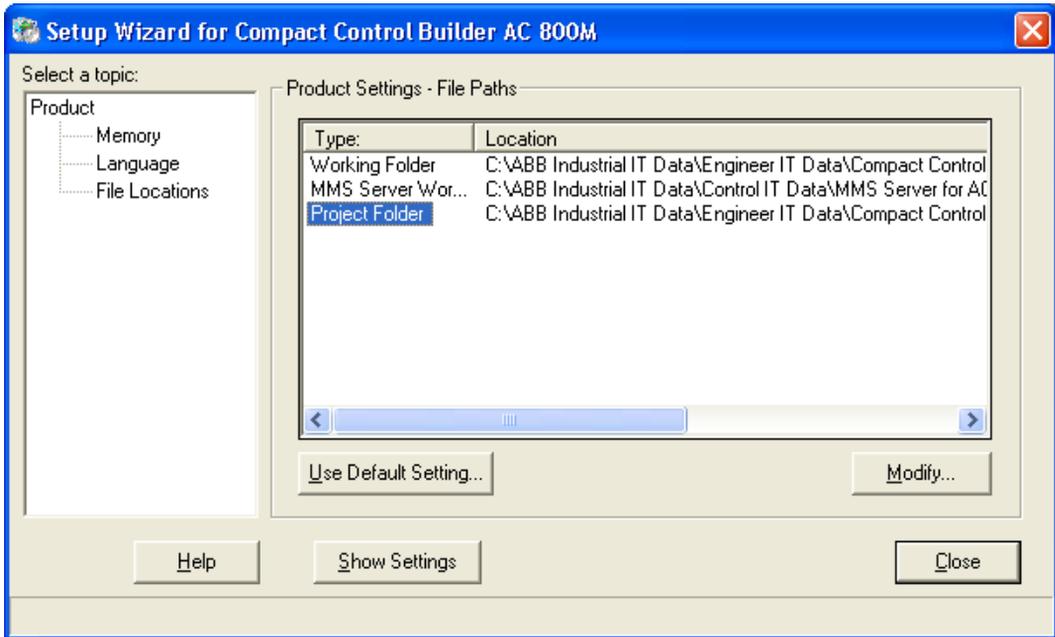


Figure 67. Product topic for file locations.

- Working folder; contains Compact Control Builder station log files, settings etc.
- MMS Server working folder; contains MMS Server log files.
- Project folder; contains projects.

Clicking *Use Default Settings*, resets the manual settings to the general default settings.



How to change the file location for the Project Folder, see [Multi-User Configuration](#) on page 98.

Multi-User Configuration

The Compact Control Builder must be installed on every PC before setting up a multi-user environment. The OPC Server can be installed on one of the Compact Control Builder stations or on a standalone PC. All PCs must be connected to the same network.

A multi-user configuration requires that all Compact Control Builder stations and an OPC Server have access to the common project files¹. During multi-user engineering, all Control Builder stations must write/read engineering changes to the common project files located in a shared project folder.

The OPC Server reads run-time data directly from the PLC over the network, by accessing the configuration data located in the common project folder. Thus, the project folder must be both shared and placed on a network server, before the team starts multi-user engineering.

Multi-user configuration consists of the following sections:

- [Creating a Shared Project Folder](#) on page 99.
- [Setting Up Compact Control Builder Stations](#) on page 100.
- [Setting Up OPC Server](#) on page 103.
- [Configuration Example](#) on page 108.
- [Guide lines for Multi-User Engineering](#) on page 110.

1. A Compact Control Builder project contains several files. These project files hold configuration data for libraries, applications, hardware, project constants etc.

Creating a Shared Project Folder

Select a PC station as the network File server (from now on MyServer). Start Windows and login with administrator role.

1. Create a project folder (from now on **AC800Mprojects**) on MyServer.
2. From **Start** select **All Programs > Accessories > Windows Explorer**. The Windows Explorer opens in MyServer.
3. Right-click **AC800Mprojects** and select **Sharing and Security** from the context menu. A Properties dialog opens.
4. Select the **Sharing** tab and click **Share this folder** radio button. Locate the settings from the Properties dialog in [Figure 68](#).

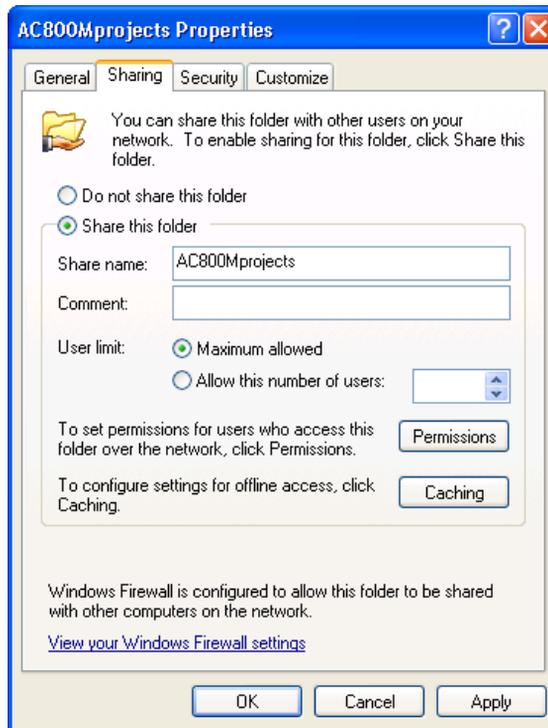


Figure 68. Property dialog with the Sharing tab active.

5. Click the **Permission** button to open the permission dialog ([Figure 69](#)).

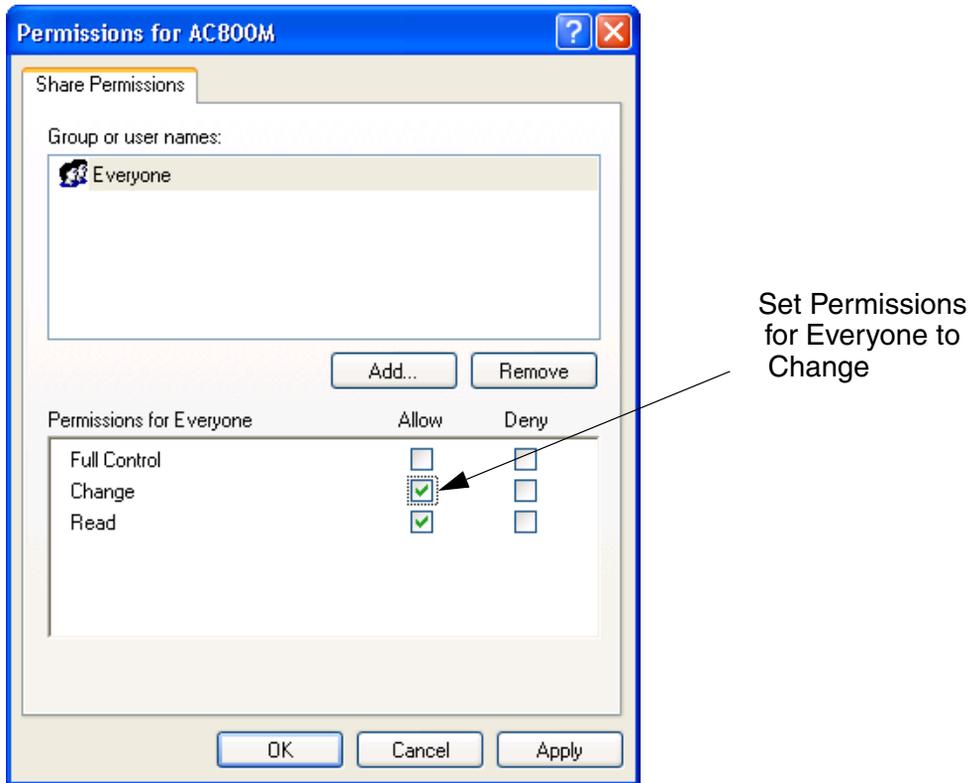


Figure 69. Share permission dialog for the shared project folder AC800Mproject.

6. Select **Change** and click **Apply**.



It is preferable to create a new user group (for example MyTeam) in Windows and add the project members to MyTeam group. Then, select the permission as **Change** for MyTeam, instead of for Everyone.

7. Click **OK** to close the Permissions dialog.
8. Click **OK** to close the Properties dialog.

Setting Up Compact Control Builder Stations

Ensure that the Compact Control Builder is installed and the PC station is connected to Ethernet. For installation instructions see [Section 2, Installing Software](#).

1. Start the **Setup Wizard** according to [Starting the Setup Wizard for Compact Control Builder](#) on page 94.
2. Select **File Locations > Project Folder**, and click **Modify** to open the Browse to Project Folder dialog.

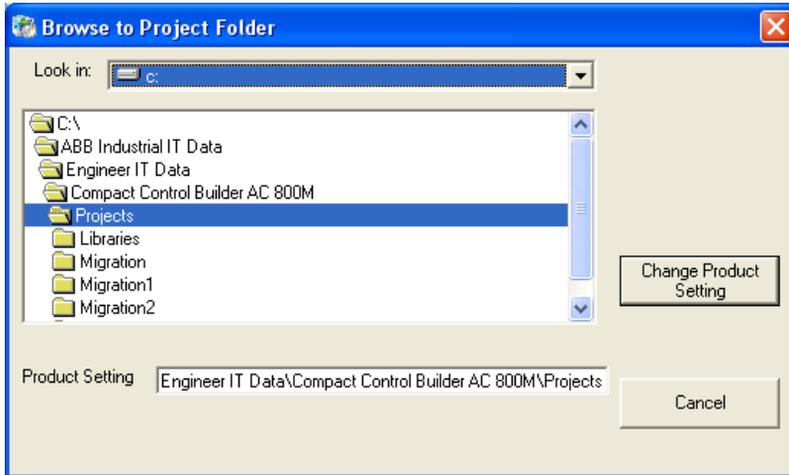


Figure 70. Browse to Project Folder dialog for locating the shared project folder



Before browsing to the (UNC) path, map a network drive in Windows. If the browse dialog does not permit browse navigation on network places, type the path in the Product Setting field (Figure 70).

The path must be specified with an UNC path that contains the server name and the shared folder name \\MyServer\AC800Mprojects.



Exclude the ABB Industrial IT Data folder and any used shared network disk from a virus scan, if the files are scanned at access. Configure the anti-virus program to scan these files and folders on demand or at a scheduled scan.

3. Locate the network File server and browse to the shared **AC800Mprojects** folder.
4. Click **Change Product Setting** to close the Browse to Project Folder dialog.

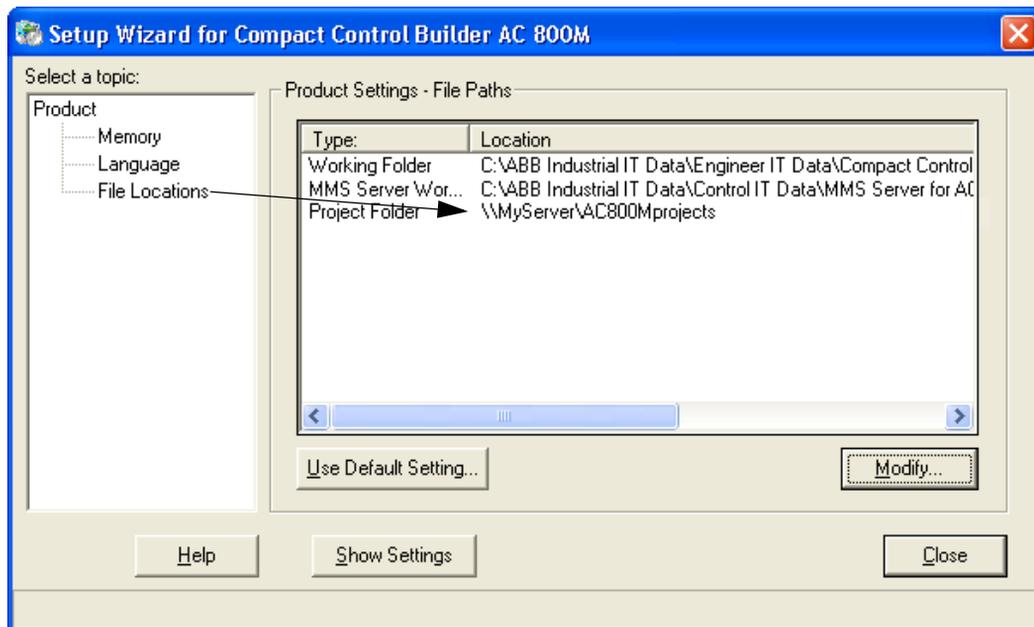


Figure 71. The new file location for the common project folder.



For more information about the differences between the Working folder and Project folder, see [File Locations](#) on page 97.

5. Click **Close** to close the Setup Wizard.
6. Repeat these steps for all Compact Control Builder stations.

Setting Up OPC Server

An OPC Server configuration for multi-user engineering requires the following:

- The path to the common project folder. This path is to be specified under the topic 'File Locations' in the Setup Wizard.
- A Service Account.

Ensure that the OPC Server is installed. For installation instructions see [Installing the OPC Server for AC 800M](#) on page 27.

1. From **Start** select **All Programs > ABB Industrial IT AC 800M > OPC Server for AC 800M 5.0> Setup Wizard**.

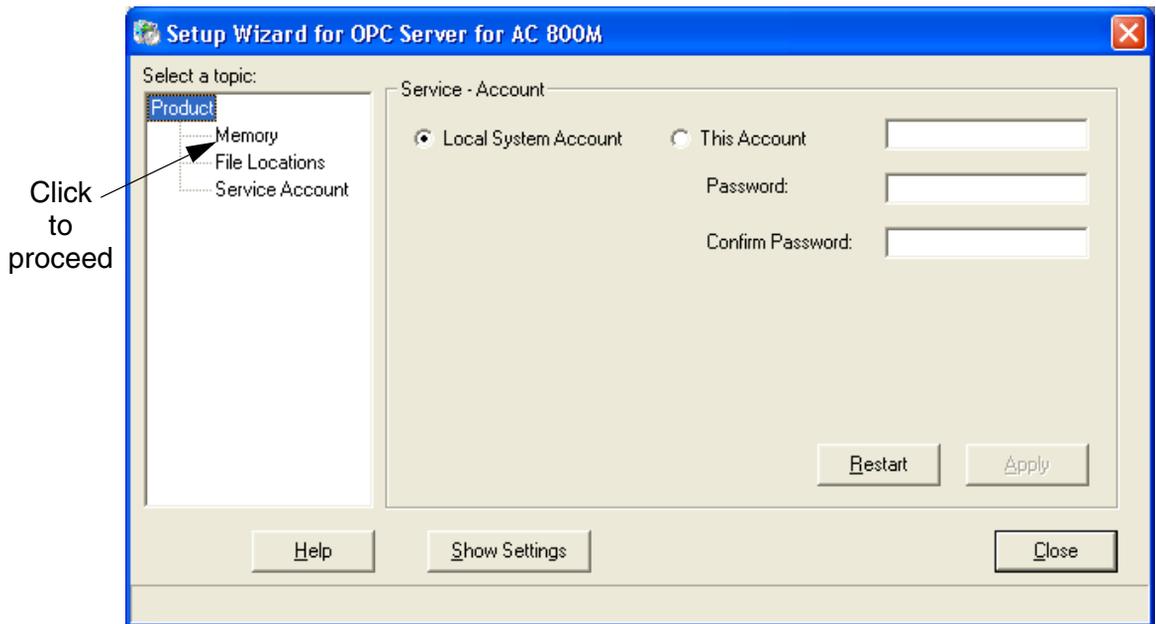


Figure 72. Setup Wizard start menu for OPC Server.

2. Select **Memory** to configure the memory settings.

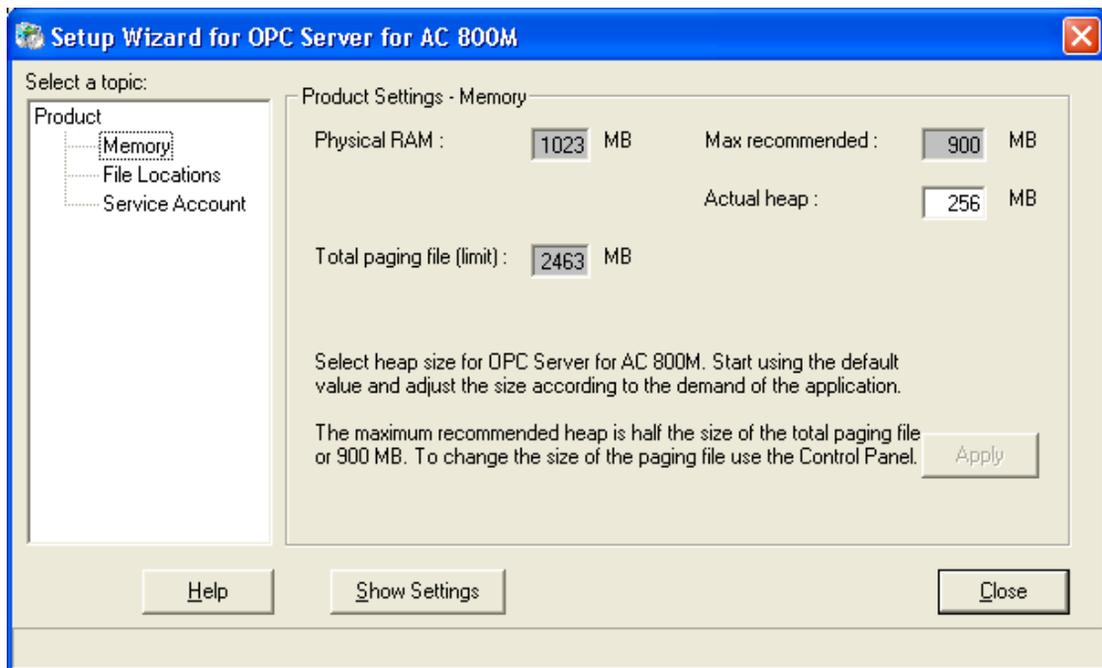


Figure 73. Memory settings for OPC Server. After configuring the memory settings, click the topic 'File Locations' to proceed.

The actual heap should be set to the size of the application but not exceed the maximum recommended heap size. Start by using the default value and then increase the value according to the application demands.

The default OPC Server heap size is 256 MB. However, in conjunction with large projects, this size is not sufficient, and it results in failure of the OPC server. If this failure occurs (if the server cannot start), increase the heap size to a value that is double the previously allocated heap size. Check the OPC Server's About box shortly after the Server is up running again, and make sure there is at least 30% spare heap.

3. Select **File Locations > Project Folder**, and click **Modify** to open the Browse to Project Folder dialog.

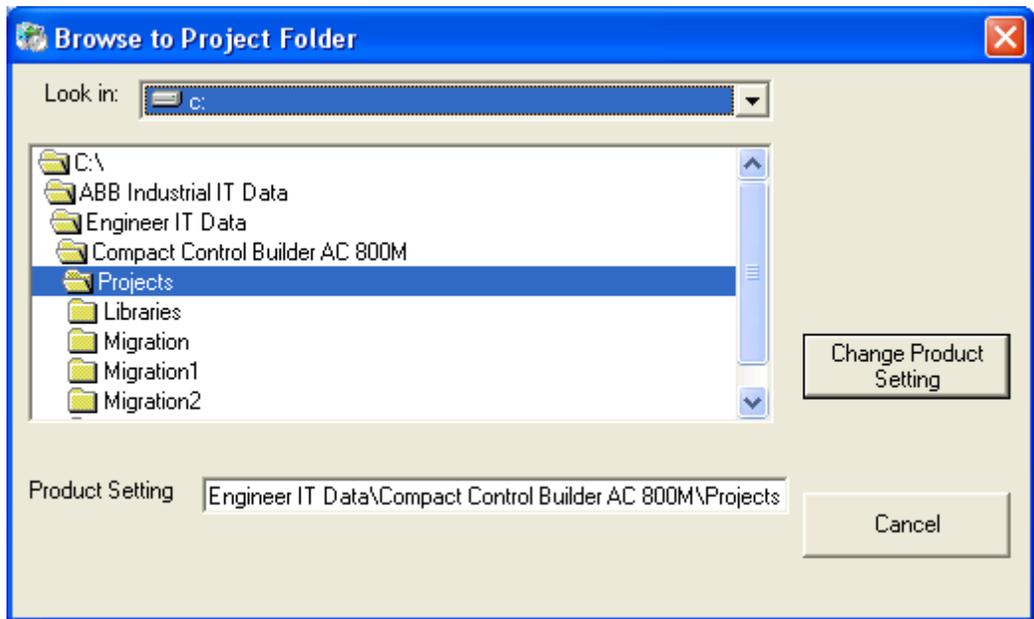


Figure 74. A Browse dialog for locating the shared project folder



Before browsing to the (UNC) path, map a network drive in Windows. If the browse dialog does not permit browse navigation on network places, type the path in the Product Setting field (Figure 74).

The path must be specified with an UNC path that contains the server name and the shared folder name `\\MyServer\AC800Mprojects`

4. Locate the network File server and browse to the shared **AC800Mprojects** folder.
5. Click **Change Product Setting** to close the Browse to Project Folder dialog.

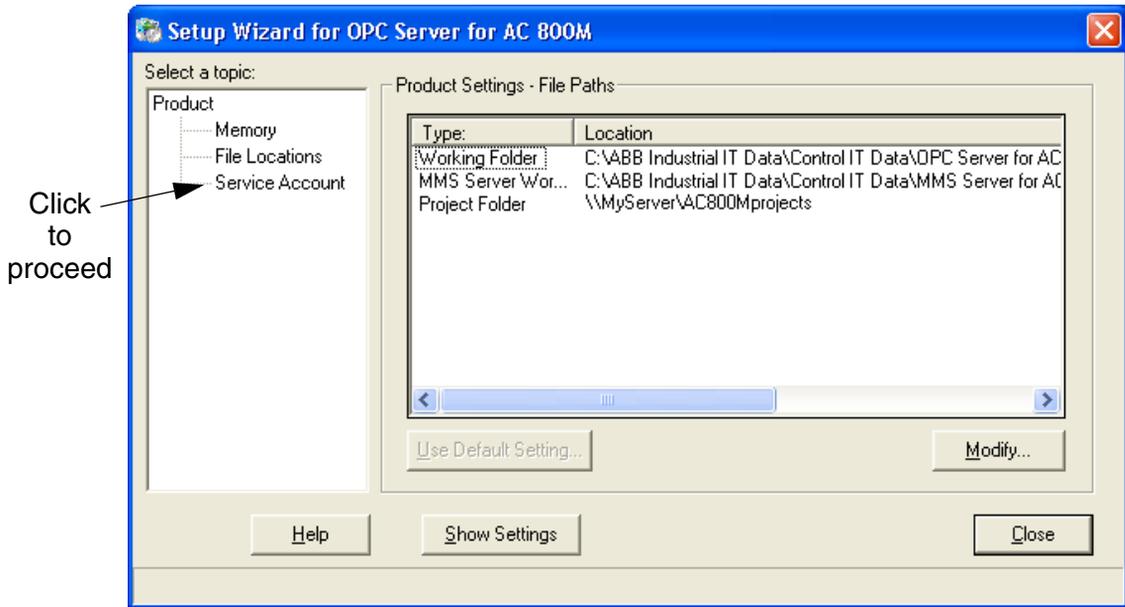


Figure 75. An example of a path to a shared project folder located on a network server. Click the topic 'Service Account' to proceed to the next step in the wizard.

6. Select the topic **Service Account**.

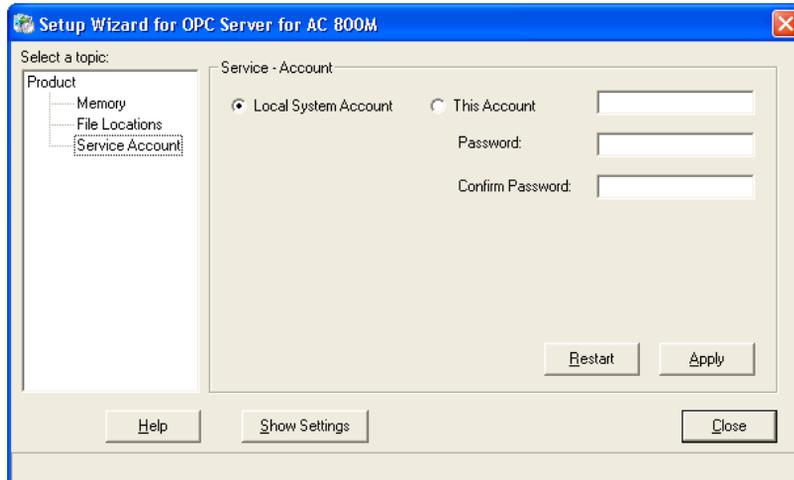


Figure 76. The Service Account Topic for OPC server.

Choosing the Service Account

The choice between '*Local System Account*' and '*This Account*' depends on the OPC Server location and the other software products that are installed on the same PC (as the OPC Server). The two account options are:

- **Local System Account** cannot read from/write to a network file server. The project folder must be created locally on the same PC as the OPC server runs. This is irrespective of the user that is logged in.
- If a network file server holds the project folder the OPC service must run with **This Account** and the User account (user, password) must have change privileges to the project folder.

Communication Failure between OPC Server and the OPC Panel

If a communication failure occurs between the OPC Server and the OPC Panel, then a pop-up menu appears with the message 'Access denied', and the OPC Panel stops responding. In such a scenario, proceed with the following steps:

- a. In Windows, search for the **OPCServerPanel.exe** file¹.
- b. Right-click **OPCServerPanel.exe** and select **Run as** in the context menu.
- c. Select the OPC Server user account (with administrator role) for running the OPC Server Panel application.

For more information about setting up user account in Windows, refer to Windows User Documentation.

7. Click **Close** to close the Setup Wizard.



Create a small project in one of the Compact Control Builder stations and verify that the project can be opened from all the other Compact Control Builder stations.

1. Normally located at C:\Program Files\ABB Industrial IT\Control IT\OPC Server for AC 800M 5.0\Bin\

Configuration Example

Assume the following network configuration:

- The OPC Server is installed on a PC together with an operator interface. The OPC Server has the Project folder path set to the File Server.
 - OPC Server copies the configuration data from the project folder to its own local working folder. It uses the configuration data to translate the live data traffic from the PLC.
 - OPC Server writes cold retain values to the shared project folder. This is one of the reasons to set the permission *Change* in Windows.
- If *Local System Account* (see Figure 76) has been selected previously in the Setup Wizard, only members of MyTeam group should login to the PC machine, or the OPC Server is interrupted.
- If *This Account* (see Figure 76) has been selected previously in the Setup Wizard, any user including an operator can login to the PC machine without interrupting the OPC Server traffic.
- Two Compact Control Builder stations with their project folder shared on the network File Server.
- One File Server with the shared project folder AC800Mprojects.
- A control system here symbolized as PLC.

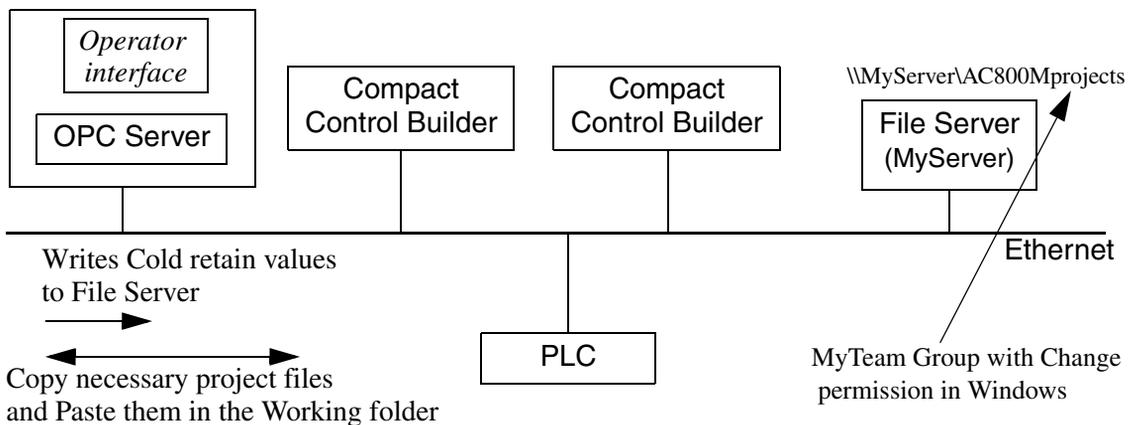


Figure 77. An example of a multi-user configuration.

Guide lines for Multi-User Engineering

A project contains a number of files, whereas every member in a multi-user environment has a decisive impact on these project files. This means that a project folder shared by several Compact Control Builders may be subjected to multiple changes at the same time.

To avoid unwanted read/write results on the project files, follow these guidelines:

1. Several members may work with different libraries etc. without difficulties, but always strive to assign one member to a specific library, application or PLC at the time.
2. If several members must work with the same library or application, then allow only one member to work with a specific Type (Program, Function block type etc.) at the time.
3. Allow only one member to work with control modules in an application at the same time.
4. If several members must work with a specific PLC, then permit only one member to work with a specific Hardware unit, Task or Access variables, at the same time.



If a rename operation affects several files, Control Builder shows the alert message and displays the corresponding files before proceeding with the rename operation.

Download

This section describes download and the checks and reports associated with download.

General Download

Select **Tools > Download Project and Go Online** to download the project in Control Builder and enter the Online mode.

Version and Online Analysis

During the version check, the project in the Control Builder is analyzed and compared with the project downloaded in the controller (if any).

The version check detects the following conditions and provide solutions:

- If the project versions are identical, and neither the application nor the controller configuration has been changed, then the project is not downloaded. The effect is the same as going online without download.
- If there is no project version mismatch, but the application or controller configuration is changed, then the changed parts are downloaded. This download depends on the next two conditions.
- If the project in the controller contains an application that is not part of the project to be downloaded, but has been downloaded as part of the same project, then this application is deleted during the download process.
- If there is another project in the controller, different from the one to be downloaded, the user must delete the project and restart the controller, before downloading the new project. See [Download New Project to Controller](#) on page 114.

The Online Analysis dialog displays the present applications and controller configurations, and whether or not they have been changed in the Control Builder. An application or controller configuration is considered changed if the version in the Control Builder is different from the version running in the controller.

Compilation

Compilation is performed in Control Builder. If any warnings or errors are detected during the compilation, a Compilation Summary dialog shows a summary of the warnings and errors. Then choose (if there are no errors) to continue or cancel the compilation.



Compiler switches can be used to set additional restrictions for the code. For more information, see the manual *3BSE035980RXXXX Basic Control Software, Introduction and Configuration*.

Change Analysis

Control Builder performs a change analysis if any of the following are changed:

- Variables, function blocks, or control modules.
- Data types, function block types or control module types.
- Libraries.
- Applications.

The change analysis is performed before downloading, to check the possibility of maintaining variable values after restart.

The change analysis detects mismatches between the application version in the controller and the application version to be downloaded.

A mismatch can occur if:

- A variable is assigned another data type.
- A variable, function block, or control module is renamed.
- A data type, function block type, or control module type is missing, renamed, or moved to another library.
- A library is given a new name (this results in a mismatch for all data types, function blocks types, and control module types from this library).
- An application has been renamed (this results in a mismatch for all data types and variables, function blocks, and control modules in the application).

For variables with attributes *Retain* or *ColdRetain*, the change analysis is performed in the following way:

1. All data types, function block types, and control module types, which existed before the change, are checked for name matching.
2. All variables, function blocks, and control modules are checked for name and type matching.

If the change analysis detects mismatches, Control Builder cannot determine how to retain variable values. A warning dialog displays information about detected mismatches. Then the mismatching names should be corrected, by giving the renamed object the new name (click **Rename** in the dialog).

Download and Go Online

The changed parts of the project (application and/or controller configuration) are downloaded to the controller(s). The controller(s) stops the running application(s), and restarts with the new/changed versions, and with variable values maintained (depending on the type of attribute and restart).

After the download is completed, Control Builder enters Online mode. In Online mode, Control Builder communicates with the controller(s), and the user can view variables and execution of the application in the controller(s) using online editors. Furthermore, the user can issue operations to the controller.



During download, if an error is detected in the downloaded controller configuration, then the message “Download aborted. See the controller log for further information.” appears. A common cause is that there is no sufficient controller memory. The details are found in the controller log. If the controller is still running, try to compile and download again. Refer to *3BSE035980XXXX, Basic Control Software, Introduction and Configuration* manual to locate the log file.

Download New Project to Controller

When **Tools > Download Project and Go Online** is selected, the Confirm Deletion of Project dialog is displayed if there is another project in the controller. To reset and restart the controller, click **Complete Reset** in the dialog. See [Figure 78](#).



Another way to reset and restart the controller is to press controller's INIT button for more than 3 seconds.

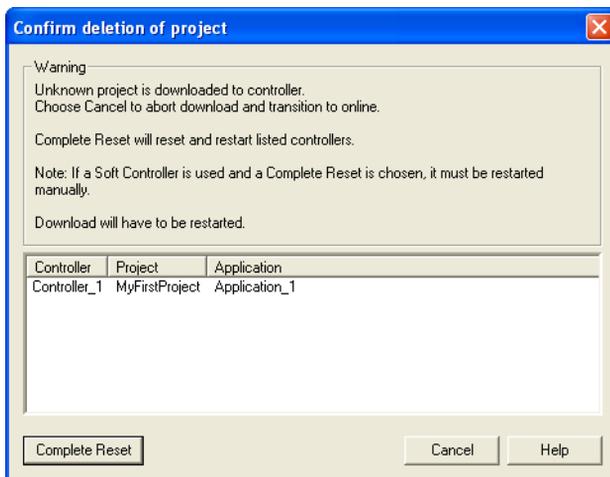


Figure 78. *Confirm Deletion of Project dialog*

The controller is reset, all existing applications in the controller are deleted, and the controller is restarted. The download of the new project can then be continued (see [General Download](#) on page 111).

Download Project to Selected Controllers

It is possible to select the controllers to download and go online, for a project that contains more than one controller. In this way, certain applications and controller configurations can be excluded from the download, and can go online with only a subset of the project to a selected controller. This reduces the compilation time.

When working in a multi-user environment, one user can work with some parts of the project, while other users can work with other parts. If the other parts of the

project cannot be compiled because they are not completed, the user can go online with parts that are completed.



If an application is connected to several controllers, it is not possible to select only one controller. The remaining controllers are added automatically.

When **Tools > Download Project and Go Online** is selected for a project with more than one controller, the Selection of Controllers dialog is displayed. For example, in [Figure 79](#), “Controller_2” and “Controller_3” cannot be separately selected or excluded since one application is connected to both these controllers.

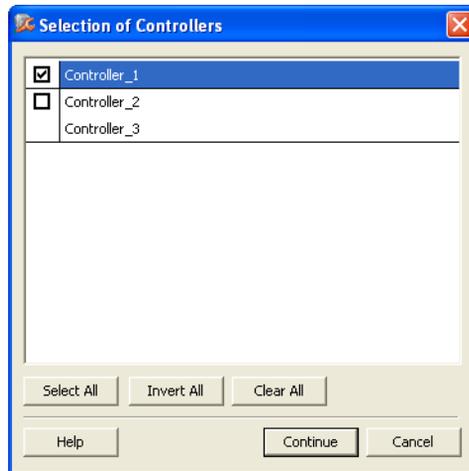


Figure 79. Selection of Controllers dialog

Another way to select a specific controller is to right-click the controller in Project Explorer, and select **Download and Online Mode**.

The Selective Download function is by default enabled, but can be disabled by selecting **Tools > Setup > Station > Application Download**, and in the Setup dialog setting the parameter *SuppressOnlineSelectionDialog* to true.

Selecting an Application to Download

Besides connecting several applications to one controller and downloading all of them, a single application can be selected for download. This is convenient when some parts, for example, in Application_1, are not ready but Application_2 is

completed and ready for testing. If both applications are connected to Controller_1, then there is an option to select if both applications should be downloaded or just Application_2.

1. Click **Continue**-button in Figure 79, an Online Analysis window opens.
2. Select the application to be downloaded. See the example in Figure 80.

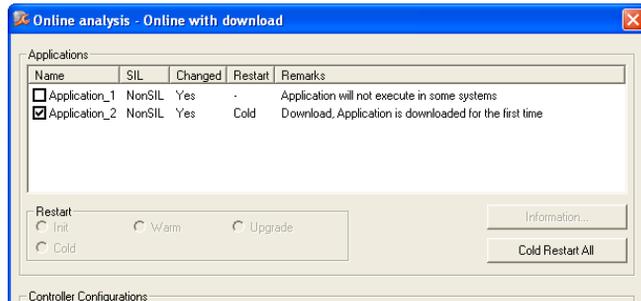


Figure 80. Connected applications to a controller. Only Application 2 will be downloaded.

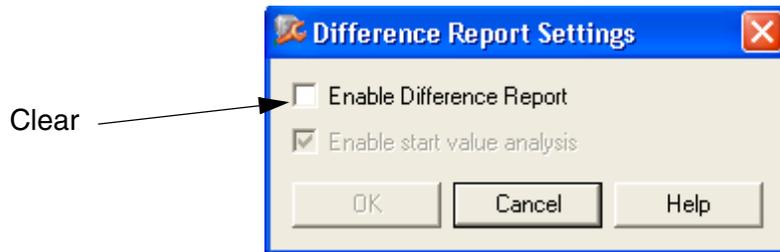
Disable/Enable Difference Report

The following instructions disables the Difference Report, thus preventing the Difference Report Settings dialog to pop-up when selecting *Test Mode*, *Online*, *Download Project* and *Go Online*.

Disable Difference Report

From the Project Explorer tree:

1. Right-click the Project icon (root level) and select **Settings > Difference Report** from the context menu. A 'Difference Report Settings' dialog opens.
2. Clear **Enable Difference Report** option.



3. Click **OK**.

Appendix B Network Redundancy



The information given in this Appendix applies only to users who intend to setup Redundant Networks.

For more information about Redundant Networks and clock synchronization, refer to the online help and the *IndustrialIT 800xA - Control and I/O, 3BSE035982RXXXX, Communication, Protocols and Design* manual (in particular, the MMS section in the manual).

Setting Up Redundant Network

The following example explains how to set up two separate redundant networks with the so called *implicit IP addressing* method. It also explains configuring IP addresses for the two PLCs and two PCs. The following sub-sections have step-by-step instructions which can be applied in the projects. After completing this example, the user must be able to add another PLC or Compact Control Builder station.

Two Separate Redundant Networks

The example consists of PLC_1 and PLC_2 on one redundant Control Network, and PC_1 with a Control Builder on another redundant Client/Server Network. PC_2, with for example an OPC Server to access the right network components, connects the two separate redundant networks according to [Figure 81](#).

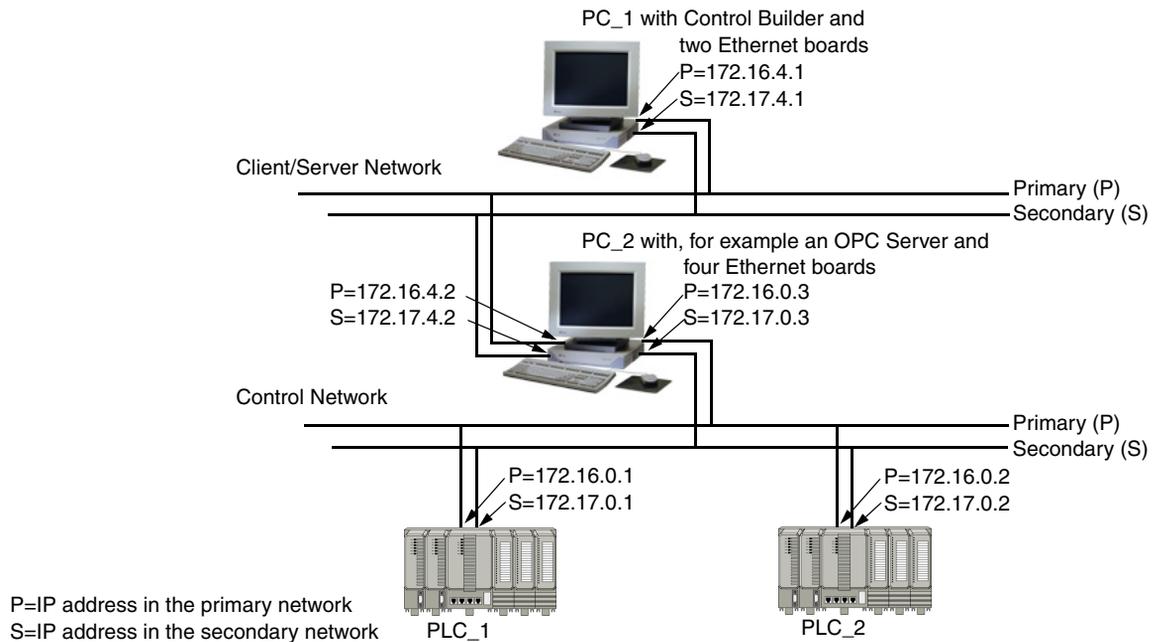


Figure 81. Redundant Control Network and redundant Client/Server Network.

Changing in RNRP Setup Wizard

Run the RNRP Setup Wizard for PC_2, otherwise the routing from PC_1, via PC_2, to the PLCs does not work.

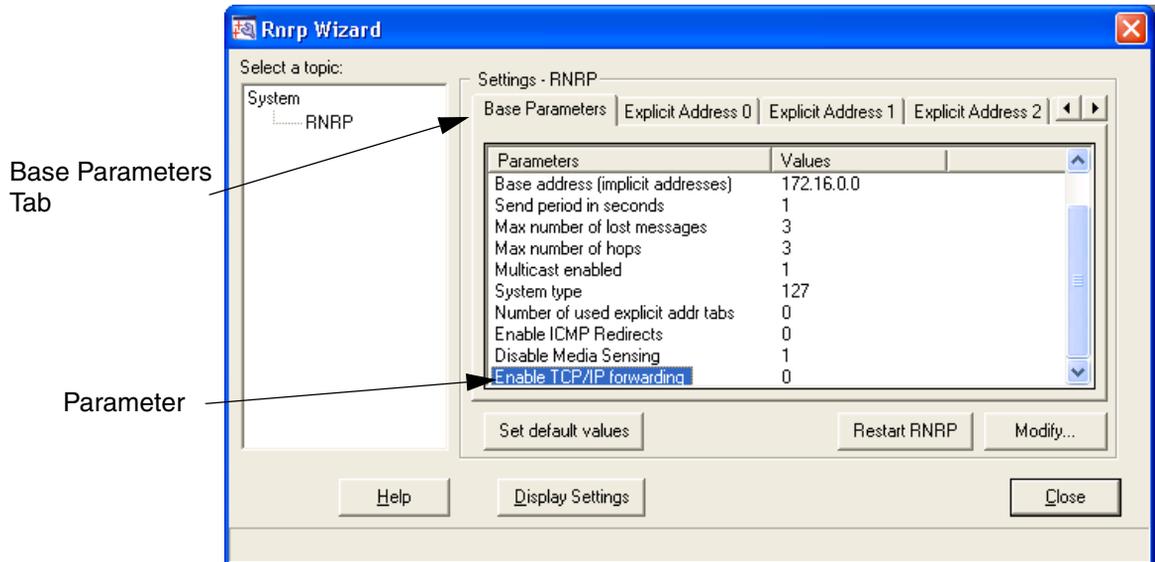
1. Right-click the **ABB RNRP**-icon located at the lower right-side of the Windows desktop. The RNRP Wizard opens.



Right-click ABB RNRP-icon

Figure 82. ABB RNRP-icon for opening the Setup wizard.

2. Make sure the **Base Parameters** tab are active.



3. Select the **Enable TCP/IP forwarding** parameter and click the **Modify** button. A value dialog opens.

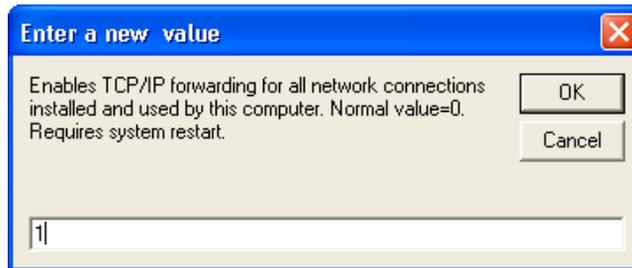


Figure 83. Parameter dialog for changing parameter values.

4. Change the parameter value to **1** instead of the default value 0 and click **OK**.
5. Click **Close** to close the RNRP Wizard.

Decide IP Addresses

First the IP addresses must be decided, and then the IP subnet mask to use for each Ethernet connection port. These redundant networks must have two IP addresses for each PLC and in PC_1 with the Control Builder. In PC_2, with the OPC Server, four IP addresses are required to the four ports in the Ethernet PC boards.

In this example, select the IP addresses displayed in [Table 5](#) below and in [Figure 81](#) as follows.

Subnet mask

Use the following subnet mask for **all** IP addresses: **255.255.252.0**.

IP addresses (X.Y.Z.Q)

- Use the recommended IP addresses in the **X.Y.** positions of X.Y.Z.Q for both redundant networks; **172.16.Z.Q** for the primary network, and **172.17.Z.Q** for the secondary network.
- Due to the subnet mask value, select the **Z.** position of X.Y.Z.Q as a multiple of four. Choose two values between; 172.16.**0.Q**, 172.16.**4.Q**, 172.16.**8.Q**, 172.16.**12.Q** etc. up to 172.16.**124.Q**. Note that the **Z.** value must also be different for Control Network and Client/Server Network.
- Select the **Q** position of X.Y.Z.**Q** as a free serial number in the range 1 - 254, for each **node** on each separate network. Thus one serial **Q** number for the primary and the secondary network ports of each PLC and PC.

Table 5. Selected settings of the IP addresses with the 255.255.252.0 subnet mask.

| Network | PLC_1 (X.Y.Z.Q) | PLC_2 (X.Y.Z.Q) | PC_1 with Control Builder M (X.Y.Z.Q) | PC_2 with OPC Server (X.Y.Z.Q) |
|---------------------------------|--------------------|--------------------|---|--------------------------------------|
| Primary Control Network | 172.16.0.1 | 172.16.0.2 | | 172.16.0.3 |
| Secondary Control Network | 172.17.0.1 | 172.17.0.2 | | 172.17.0.3 |
| Primary Client/Server Network | | | 172.16.4.1 | 172.16.4.2 |
| Secondary Client/Server Network | | | 172.17.4.1 | 172.17.4.2 |

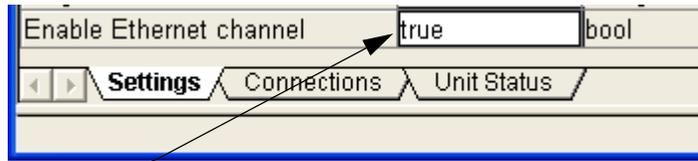
Setup Using the IPConfig Tool

To get initial access to the PLCs, assign them with a first primary IP address.

1. Connect a PC, running the program *IPConfig* tool, to the PLC_1 Tool port (via a serial cable).
2. Follow IPConfig online help instructions and set the primary IP address to 172.16.0.1 (see [Table 5](#)).
3. Repeat Step 1 and Step 2, for PLC_2.

Configure PLC Ports from Project Explorer

1. In the Project Explorer of PC_1, configure the project with PLC_1 and PLC_2.
2. In PLC_1, double-click Ethernet port number 1 to open its editor.
3. In the editor Settings tab, set the IP address parameter to 172.16.0.1 (see [Table 5](#)), and the IP subnet mask to 255.255.252.0. No other parameter setting is required.
4. Repeat step 2 and 3 for Ethernet port number 2 of PLC_1, set its IP address to 172.17.0.1 (see [Table 5](#)).
5. From Ethernet port number 2, select **Settings** tab and set the **Enable Ethernet channel** parameter to true ([Figure 84](#)).



Change the parameter from false to true

Figure 84. Hardware editor for Ethernet port No. 2 in Project Explorer. The parameter must be true for Network redundancy.

6. Repeat step 2 to 5 for PLC_2.

Configure PC Ports in Windows XP Professional

Configure the Ethernet board ports for PC_1 and PC_2, using the following steps:

1. In PC_1, select **Start>Control Panel**.
2. Select **Network and Internet connections**.
3. Select **Network Connections**.
4. In Network Connections, right-click Local Area Connection and select **Properties**. The Local Area Connection Properties dialog is displayed.
5. In the Connect Using list, select the Ethernet board that is to be connected to the primary network.
6. Select **Internet Protocol (TCP/IP)** and click **Properties**. The Internet Protocol (TCP/IP) Properties dialog is displayed.
7. Select **Use the following IP address** and enter the IP address (172.16.4.1 for the primary network in PC_1, see [Table 5](#)), and the subnet mask 255.255.252.0. Click **OK**.
8. Repeat steps 1 to 7 for the secondary network (using the IP address 172.16.4.2).
9. Click **OK**.
10. For PC_2, repeat steps 1 to 9 for **all four** Ethernet boards to be configured.
11. Click **OK**. The PC IP addresses and subnet masks have now been set for all Ethernet ports connected to the two networks.

Download Project and Go Online

When all IP addresses and subnet masks are set, download the project and go online in the Control Builder. After a while, redundant network communication is enabled.

Setting Clock Synchronization using the CNCP Protocol

This subsection describes how to setup a clock synchronization using the CNCP protocol. CNCP offers relative clock synchronization down to milliseconds, which means that the real time clocks in the PLCs will never be more than a few milliseconds apart. Hence, absolute time will not necessarily be totally correct. See also the *IndustrialIT 800xA - Control and I/O, 3BSE035982RXXXX, Communication, Protocols and Design* manual for other available clock synchronization methods.



CNCP requires that RNRP is properly configured in the PLCs.

Configure the clock master

Assign one PLC to be the clock master (any type of PLC is adequate). From the Project Explorer tree:

1. Right-click the assigned PM unit and select **Editor** from the context menu. The hardware editor opens.
2. Select the **Settings** tab and locate the following parameters.
3. Set the following (four) parameters according to [Figure 85](#).

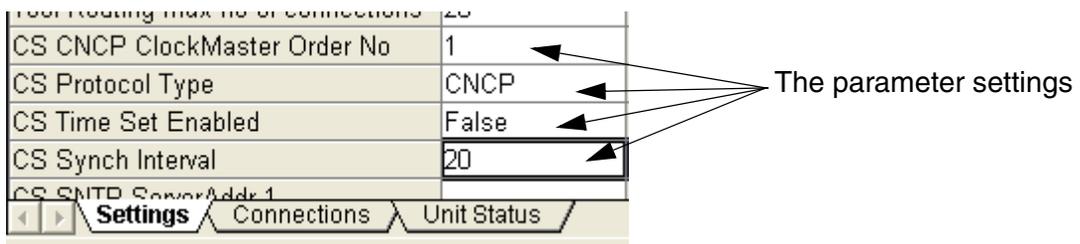


Figure 85. The parameter settings under the Settings tab in the hardware editor.

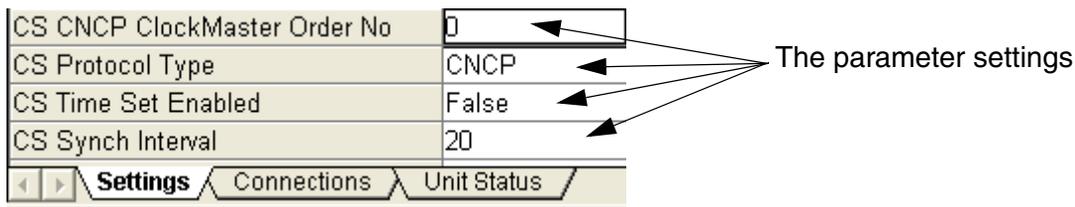
4. Save the settings and close the editor (**Ctrl+U**).

5. Perform a download to the (clock master) PLC.

Configure the clock slaves

Assign all the other PLCs to be clock slaves (they should all be configured in the same way). From the Project Explorer tree:

1. Right-click the PM unit and select **Editor** from the context menu. The hardware editor opens.
2. Select the **Settings** tab and locate the following parameters.
3. Set the following (four) parameters according to [Figure 86](#).



| | |
|------------------------------|-------|
| CS CNCP ClockMaster Order No | 0 |
| CS Protocol Type | CNCP |
| CS Time Set Enabled | False |
| CS Synch Interval | 20 |

The parameter settings

Figure 86. The parameter settings under the Settings tab in the hardware editor.

4. Save the settings and close the editor (**Ctrl+U**).
5. Perform a download to the (clock slave) PLC.

Design

IP Address

A communication channel IP address is a 32-bit word (4×8 bits) that can be entered as a string X.Y.Z.Q of four decimal numbers 0-255, separated by periods. The IP standard uses the terms *NetID* and *HostID*. The *subnet mask* specifies the boundary between the NetID part and the HostID part of the IP address (the zero bits indicate the HostID part). Depending on the value of X, IP addresses are divided mainly into three classes, A–C:

Table 6. IP address classes.

| Class | Value of X | NetID | HostID | Possible host IP addresses | Default subnet mask |
|-------|------------|-------|--------|----------------------------|---------------------|
| A | 1-126 | X | Y.Z.Q | X.0.0.1-X.255.255.254 | 255.0.0.0 |
| B | 128-191 | X.Y | Z.Q | X.Y.0.1-X.Y.255.254 | 255.255.0.0 |
| C | 192-223 | X.Y.Z | Q | X.Y.Z.1-X.Y.Z.254 | 255.255.255.0 |

The *Redundant Network Routing Protocol (RNRP)* developed by ABB handles alternative paths between nodes and automatically adapts to topology changes. RNRP uses more terms than the standard IP, namely *network area* and *node number*. By selecting 225.255.252.0 as the subnet mask, the last 10 bits constitute the node number (i.e. host ID, 0-1023). Note that the largest permitted node number is 500. The remaining NetID part is used for network ID (16 bits), local flag (1 bit), and network area number (5 bits). The last two bits of the network ID make up the path number, where 0 indicates the primary network and 1 the redundant secondary network.

Consequently, RNRP requires a different interpretation of the IP address to the IP standard.

It is recommended that the RNRP interpretation of the IP address be used. If the decimal numbers are converted to binary numbers, the address can be interpreted as follows:

XXXXXXXX.XXXXXXPP.LAAAAANN.NNNNNNNN

Each position represents a binary digit (0 or 1). The different parts of the address signify the following:

Table 7. IP address converted into binary.

| Binary number | Number of bits | IP term | RNRP term |
|-------------------|----------------|---------------|---------------------|
| XXXXXXXX.XXXXXXPP | 16 | Network ID | Network ID |
| LAAAAA | | Subnetwork ID | |
| PP | 2 | | Path number |
| L | 1 | | Local flag |
| AAAAA | 5 | | Network area number |
| NNNNNNNNNN | 10 | Host ID | Node number |

The subnet mask sets the boundary between the host ID part and the subnet ID part and is selected as 11111111.11111111.11111100.00000000 (=255.255.252.0 in decimal notation).



The network ID must be identical for all nodes on the same network. It is recommended that an address be selected from the private IP address space, which has the following advantages:

- There is no requirement to apply to the licensing authorities for an IP address.
- Some protection is gained against illegal access, since private addresses are not permitted on the public Internet.
- The firm connection between IP and RNRP parameters reduces the risk of inconsistency.



The following primary network IP addresses are recommended (class B):

172.16.0.0
172.20.0.0
172.24.0.0
or 172.28.0.0

By converting the second decimal number into binary, the path number is converted to 0 (see also [Table 5](#) and [Table 7](#)).

Example 1:

Convert the subnet ID 11111111.11111111.11111100.00000000 to decimal.

By writing the first part in groups of four binary digits (1111), the hexadecimal equivalent is FF. The decimal equivalent is 255. The second part is identical, that is, 255. The third part 1111 1100 equals FC (hexadecimal) and 252 (decimal). The fourth part equals 0 in both decimal and hexadecimal notation. Consequently the subnet ID is written 255.255.252.0 in decimal notation.

Example 2:

IP address 192.168.255.25 written in binary notation:

It can be written C0.A8.FF.19 in hexadecimal form, which means 11000000.10101000.11111111.00011001 in binary notation.

Separating Client/Server and Control Network

The Control Network must be protected from foreign traffic that can be a security risk and also cause undesired load on both the nodes and network. To avoid these risks the Control Network should be physically separated from the Internet and protected by servers and/or fire walls. In large configurations such separation may also be desirable between the Control Network and client/server networks.

Client Node and Server Node

In a large network topology, only nodes with real-time communication requirements may be connected to the Control Network. *Client nodes* (such as engineering stations, operator stations, etc.) not belonging to the Control Network may go

through a *server node* that performs authority control and can disable routing (transfers) to nodes on the Control Network.

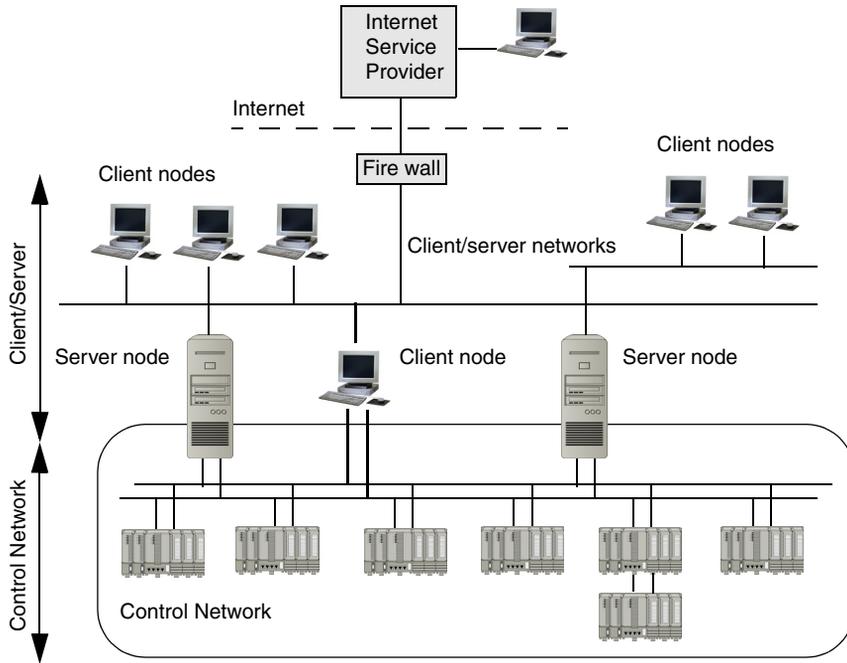


Figure 87. Separation of Control Network and Client/server networks.

It is, however, possible to have client nodes such as operator stations and engineering stations connected directly to the Control Network. If such a client needs access to the client/server network, a separate network connection is possible.



Traffic other than that between Compact Control Builder products may jeopardize performance.

Only IP traffic generated by Industrial^{IT} products is allowed to reach the PLCs on the Control Network.

Summary of Configuration Steps

1. Specify network ID (IP address and subnet mask).
2. Split the control network into network areas.
3. Specify local network areas.
4. Specify network area numbers.
5. Assign path numbers.
6. Decide which devices should run the routing functionality.
7. Assign IP addresses to each device.
8. Assign redundant functionality to each system.

Appendix C Upgrade

This appendix describes how to upgrade Compact Control Builder (4.x) projects, and Control Builder M (3.2/x) projects to a Compact Control Builder (CCB) 5.y project version.

The boxes in Figure 88, illustrates different possible upgrade options. For example, the projects from Advant Control Builder and Control Builder M (3.0 and 3.1) must first be upgraded to Control Builder M (3.2/x). In order to do so, refer to the corresponding upgrade section in Release Notes for Control Builder M (3.2/x) and follow the instructions provided to upgrade to 3.2/x. Once they are upgraded to (3.2/x), proceed with the upgrade instructions provided in this appendix.

However, upgrading Compact Control Builder from 5.x to 5.y is an easy operation. Install the latest Compact Control Builder (5.y) products and re-open the projects, and they are immediately upgraded with the latest features. There is no need to remove the (5.x) products from the Control Panel (*Remove Programs*).

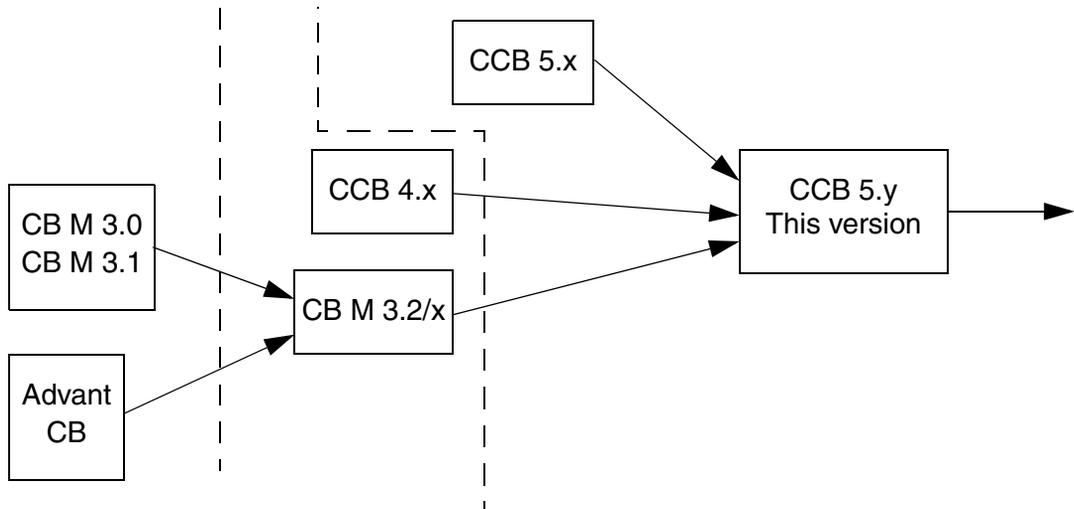


Figure 88. Only upgrade from CB M (3.2/x) and CCB (4.x) to CCB 5.y is covered in this appendix.

Installation Considerations before Upgrade

A typical installation of Compact Control Builder installs the libraries that are needed for new projects.

Some libraries from the previous version are needed during an upgrade. These libraries are installed if any of the two following options are enabled through the Custom Installation option during the Control Builder installation:

- AC 800M Coexistence 5.0.1.

This feature must be enabled for upgrade from versions 5.0.0/x or 5.0.1/x.

- AC 800M Classic Libraries.

This feature must be enabled if the project that needs to be upgraded, uses any of these classic library versions:

- CommunicationLib 1.0
- ProcessObjBasicLib 1.0
- ProcessObjExtLib 1.0
- SerialLib 1.0
- SerialCommLib 1.x

If these features were not enabled during installation, then the existing Control Builder installation must be modified before performing the upgrade. See [Modifying Control Builder Installation](#) on page 134.

Modifying Control Builder Installation

To modify the existing Control Builder installation to install the required libraries before the upgrade:

1. Go to **Start > Settings > Control Panel > Add or Remove Programs**, and select the installed version of ABB Compact Control Builder AC 800M.
2. Click **Change** to open the Welcome screen of the Compact Control Builder AC 800M Setup Maintenance program.

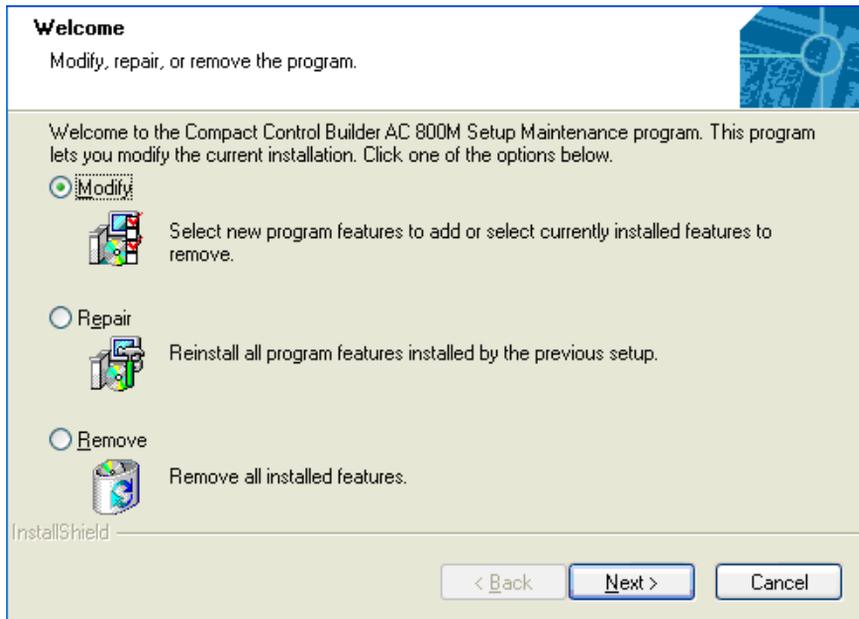


Figure 89. Welcome screen of the Compact Control Builder AC 800M Setup Maintenance program

3. Click the **Modify** radio button.
4. Click **Next** to open the Select Features screen.

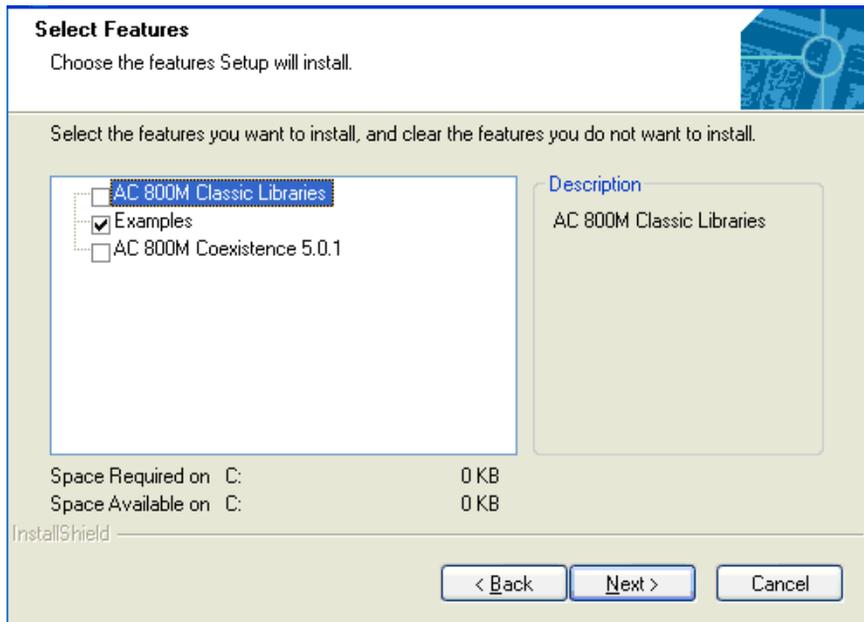


Figure 90. Select Features screen of the Compact Control Builder AC 800M Setup Maintenance program

5. Select the features that need to be installed. See [Installation Considerations before Upgrade](#) on page 134.
6. Click **Next** to modify the installation.

Introduction

Compact Control Builder AC 800M comes with features and performance similar to Control Builder M Professional 3.2 version.

An upgrade involves the following:

1. Licences, see [Licenses](#) on page 137.
2. Products, see [Products](#) on page 138.
3. Applications, see [Applications](#) on page 139.



Upgrading requires the plant to be shut down. To guarantee the functionality of the upgraded system, follow these upgrade instructions carefully.

Licenses

Contact the sales representative to perform the license upgrade. Collect all Industrial IT licenses and have them available when contacting ABB. [Table 8](#) lists and describes the Control Builder license structure.



Upgrading from 4.x to 5.y does not require any additional purchased licenses.

Table 8. License Structure between version 3.2/x and 5.y

| License in Version 3.2/x | License in Version 4.x/5.y |
|----------------------------------|---------------------------------|
| Advanced Process Control license | Obsolete - now included in CPU. |
| Basic Control Software license | |
| Binary Control Software license | |

Table 8. License Structure between version 3.2/x and 5.y (Continued)

| License in Version 3.2/x | License in Version 4.x/5.y |
|----------------------------------|--|
| Control Builder M Basic | Compact Control Builder for AC 800M ⁽¹⁾ |
| Control Builder M Professional | |
| Control Builder M Standard | |
| OPC Server for AC 800M/C license | |
| OPC Server for AC 800M/C license | OPC Server for AC 800M |

(1) The licenses from version 3.2/x can be upgraded to a single license when purchasing a Compact Control Builder license version 5.0. Furthermore, all the product licenses that were collected under the Compact Control Builder license in version 4.x are fully supported in the 5.y version.

Products



All PLCs, Control Builder and OPC Server nodes must be shut down during the upgrade. A plant shutdown is needed, because a software upgrade of the PLC firmware and control software stops PLC execution.

Table 9 lists and describes specific considerations to take into account before and after upgrading.

Table 9. Product Upgrade Considerations

| Issue | Considerations |
|---------------|--|
| Compatibility | Control Builder, AC 800M Firmware, and OPC Server for AC 800M must all be upgraded to the new version in order to work together. However, PLC peer-to-peer communication is possible with other PLCs running version 2.x, 3.x or 4.x by means of Access Variables. |

Table 9. Product Upgrade Considerations

| Issue | Considerations |
|---|--|
| Privilege Handling | All privilege/user handling done in Control Builder M in the previous version (PrivUsers/PrivClasses) will be lost during the upgrade. The Compact Control Builder does not have any privilege handling. |
| AC 800C and Advant Controller 250 is no longer supported. | Projects including these controller types cannot be upgraded. |

Applications

Saving Application and Configuration Data

The data described under the following subsections should be saved on a safe media before upgrading.

Settings



Both 3.x and 4.x users must read this upgrade subsection

It is very important to save Cold Retain values from the previous version on files, before beginning the upgrade.

1. From Control Builder Menu bar, select: **Tools > Save ColdRetain Values**



Control Builder must be in the Online mode in order to use the **Tools > Save ColdRetain Values** menu command.

2. Repeat the procedure for each Control Builder project.

Failure to make new files can cause some or all Cold Retain marked variables to revert to older values after upgrading.

3. Record the memory setting for OPC Server and Control Builder found in the Setup Wizard for each product.
4. From the OPC Server Panel, select **File > Save Configuration**. The OPC configurations will be saved.

Projects



Only 3.x Users must read this upgrade subsection.

Projects created with Control IT for AC 800M/C Version 3.2/x (or later 3.2 version) can be used after the upgrade; however, in some cases minor modifications must be made.

- **Save Control Builder M Projects as Official Versions:** Before upgrading a project, save the entire project as an official version. Unofficial parts of a project will not be upgraded. If the projects contain user-made HWD-files, they must also be saved. Save the Control Builder M project directory on a safe media; include new Cold Retain values files.
- A single library can not be upgraded by itself. To upgrade a library, include it in a dummy project.
- Acquired libraries that are not delivered together with Control Builder M should normally not be upgraded. Contact the vendor of the acquired libraries to get an updated version.

OPC Server for AC 800M



Both 3.x and 4.x Users must read this upgrade subsection.

Copy the following OPC Server files from the OPC Servers to safe media:

- systemsetup.sys, located in:

```
... \ABB Industrial IT Data \Control IT Data \
  OPC Server for AC 800M n.n
```
- Configuration files (*.cfg), located in:

```
... \ABB Industrial IT Data \Control IT Data \
  OPC Server for AC 800M n.n \Files
```

Control Builder M



Only 3.x Users must read this upgrade subsection.

The procedure for backing up Control Builder M projects is described in *Control Builder Online Help*. Choose to make a complete backup.

Standard libraries, and all other libraries, stored in the Control Builder M standard library path, typically:

```
... \Program Files\ABB Industrial IT\Engineer IT\  
Control Builder M [variant] 3.2\Libraries
```

or below, are not included in a project backup. The standard libraries will exist in new versions in the new system. Other libraries in that path or below must be handled manually. The best method is to save them to another path before making the backup.



Do not compress the backup.

Remove Products



Both 3.x and 4.x Users must read this upgrade subsection.

Stop third party OPC Clients and shut down Control Builder on all nodes.

In order to upgrade, the previous version or revision must be **removed** before installing the new version. If several products (that is, Control Builder and OPC Server, SoftController, User Documentation, Firmware and RNRP) are installed on the same PC, all should be removed before the new versions are installed.

The following services must be manually stopped before the products are removed/updated:

- MMS Server for AC 800M.
- OPC Server for AC 800M.
- RNRP Service.



It also applies for any other service that automatically starts any of above services.

The RNRP Service should be stopped in the Windows Services Overview, which is found at **Start > Control Panel > Administrative Tools > Services**.



No existing projects will be deleted from disc when a previous Control Builder version is removed. The new Control Builder will be able to upgrade the projects left on disc.

Install Products



Both 3.x and 4.x Users must read this upgrade subsection.

Install the products by follow the instructions given in [Section 2, Installing Software](#), (this manual).

Restore Application and Configuration Data

OPC Server for AC 800M



Both 3.x and 4.x Users must read this upgrade subsection.

The memory setting for OPC Server needs to be set from the Setup Wizard. This is the value recorded under [Saving Application and Configuration Data](#) on page 139.

1. Manually restore the previously saved systemsetup.sys file to the following location:

```
ABB Industrial IT Data\Control IT Data\OPC Server for  
AC 800M
```

Restore the configuration files (*.cfg) to the Files folder in the same location.

Compact Control Builder



Both 3.x and 4.x Users must read this upgrade subsection.

Do not start Compact Control Builder until Step 3.

The memory setting for Compact Control Builder needs to be set in the Setup Wizard. This is the value recorded under [Saving Application and Configuration Data](#) on page 139.

Saved Libraries stored on the Control Builder M standard library path, typically:

```
...\Program Files\ABB Industrial IT\Engineer IT\  
Control Builder M [variant] 3.2\Libraries
```

will not be found during the conversion procedure. Perform the following steps before the conversion if such libraries exist.

Move the libraries to the folder:

```
...\Program Files\ABB Industrial IT\Engineer IT\  
Compact Control Builder AC 800M\Libraries
```

A library will be converted only once. It is possible to upgrade several projects that include the same libraries, but the libraries will only be converted during upgrade of the first project.

1. Copy the Control Builder M projects:
 - a. Setup the project directory.
 - b. If the projects are retained on the hard drive after removing a previous Control Builder version, then upgrade from that location without moving the projects. If the projects are restored from a backup diskette then copy the projects to a directory located directly in the root directory on the hard drive. Example: **C:\MyTempUpgrade**
Include all Cold Retain value files and any user-made HWD-files.



It is not recommended to do an upgrade if the projects are copied to the Compact Control Builder project directory. The default Compact Control Builder project directory is:

```
...\ABB Industrial IT Data\Engineer IT Data\  
Control Builder AC 800M\Projects
```

- c. Make sure that all files in the project directory are not marked as Read-only which can happen if the files have been stored on a CD-ROM.
 - Use Windows Explorer to select the folder.
 - Right-click **Properties** to open the Properties dialog.
 - Uncheck the **Read-only** check box in the Attributes frame.
 - When the Confirm Attributes Changes dialog appears, select **Apply changes to this folder, subfolders and files**.
 - Click **OK** in the Confirm Attributes dialog and then again in the Properties dialog.
 - d. (**Only valid for 3.2/x**) If the projects contain S900 I/O, they must be converted with the S900 I/O Conversion tool. Refer to the S900 I/O Parameter Name Changes issue in [Table 12](#).
2. Upgrade the Control Builder project.

Libraries, Applications, and Controllers originally placed in folders different from the project folder will not be upgraded if the project backup on safe media are restored using Windows Explorer.



The CI851 and CI854 HWD-files will be automatically placed in hardware libraries after the upgrade, unless the hardware libraries do not contain these files already.

The following instruction addresses users who wish to upgrade their projects from Control Builder 3.2/x and from Compact Control Builder 4.x.

3. From the Compact Control Builder select **Tools > Maintenance > Upgrade Project**. An Upgrade Project context menu will open.
4. Select project version from which the upgrade is required (Control Builder 3.2 or Compact Control Builder 4.1 and Later).
5. Browse to the project file (*.prj) and click **Open**. A 'Save As' dialog will open.
6. Select location and click **Save** in the project folder.
7. Wait for the upgrade to finish.
 - a. During the upgrade process, some messages may be displayed. Read the messages and choose among the available options.

- b. The upgrade may take up to 30 minutes. The Compact Control Builder may stop responding during that time. If the process is interrupted, it must be restarted. The Windows Task Manager can be used to supervise the completeness.



During the upgrade, if the mouse pointer, when moved over the Compact Control Builder Project Explorer, indicates it is busy, then the upgrade is still in progress.

8. Repeat [Step 3](#) to [Step 7](#) for all other projects.

Upgrading from SV5.0 to SV5.0 SP2

Upgrading SV5.0 to SV5.0 SP2 is performed by any one of the methods below:

- Open the project by selecting open project in CCB.
- Select **Tools > Maintenance > Compact Control Builder 4.1 and Later** option.

While upgrading, an option to replace the BasicHWLib with the latest version appears. When the BasicHWLib is replaced, the firmware in the controllers also needs to be upgraded.

Upgrade PLC Firmware



Both 3.x and 4.x Users must read this upgrade subsection.

Upgrade PLC firmware to the new version in order to be compatible with other products of version 5.0. Firmware in all communication interfaces must also be upgraded to the latest version.

Handling a download



Before handling a download, refer to the compatibility issues given in [Table 10](#), [Table 11](#), and [Table 12](#).

1. Open the project.
2. Ignore the following warning messages that may be displayed in the message pane of the Project Explorer when opening the project.
 - At line xxx: Unknown parameter name: xxx

- At line xxx: Parameter type mismatch: xxx

Make a dummy change in the hardware configuration for all hardware units if such messages are shown.

3. Go online with *download* and answer any mismatch dialog with the corresponding new name, in order to retain any Cold Retain values (for example, settings of PID loops).



It is very important to complete the mismatch handling during the first download. If not, some or all Cold Retain marked variables relinquish their saved values. See [Handling Mismatch for cold retain values](#) on page 146.

Handling Mismatch for cold retain values

Upgrade to version 5.0 may give mismatch for cold retain values.

- For example, the 3.2/x library SystemLib is nowadays divided into a System folder and a Basic library.
- Furthermore, the CommunicationLib from 3.2/x has been divided into several communication libraries (for example, MMSCommLib, ModBusCommLib, and so on).

This means that Compact Control Builder will display the version mismatch dialog when downloading the project after upgrading to version 5.0. To prevent that cold retain values are lost during upgrading, examine the solutions provided below.

Solutions:

- An Application not found mismatch dialog may warn for the Source code unit SystemLib when an upgraded project is downloaded for the first time. Click **Rename** in the dialog and choose to rename the SystemLib to BasicLib. If this does not work, then the application does not have any type that has been moved to BasicLib. Cancel the Rename dialog and select **Next Mismatch**. Refer to Control Builder Online Help for additional information.
 - If the version mismatch dialog displays 'CommunicationLib not found', rename the old CommunicationLib to any of the newly divided communication libraries in the version mismatch dialog.
4. From the hardware tree in Project Explorer, select the PLC and right-click **Properties > Heap Utilization**. A Heap Utilization dialog opens.

5. Check the spare memory needed for online changes in each PLC.
6. Repeat these steps for all other projects.

Adjustment of library path



Only 3.x Users must read this upgrade subsection.

The following manual adjustments to the project file (*.prj) is needed in order to upgrade such projects correctly. Delete all directory references, other than those to *Libraries:*.

Before adjustments:

```
FileUnits
( Application
  ( Name 'Application_1'
    Directory '' )
  Library
  ( Name 'AlarmEventLib'
    Directory 'Libraries:\' )
  Library
  ( Name 'IconLib'
    Directory 'Libraries:\' )
  Library
  ( Name 'MyOwnLibrary'
    Directory '..\..\MyOwnLibs\' )
ControlSystem
( Name 'Controller_1'
  Directory '' ) )
```

After adjustments:

```
FileUnits
( Application
  ( Name 'Application_1'
    Directory '' )
  Library
  ( Name 'AlarmEventLib'
    Directory 'Libraries:\' )
  Library
  ( Name 'IconLib'
    Directory 'Libraries:\' )
  Library
  ( Name 'MyOwnLibrary'
    Directory '_' )
  ControlSystem
  ( Name 'Controller_1'
    Directory '' ) )
```

Compatibility Issues

SV5.0 SP1 to SV5.0 SP2 Compatibility Issues

[Table 10](#) lists the Control Builder compatibility issues, and the suggested solutions, when upgrading from SV5.0 SP1 to SV5.0 SP2.

Table 10. SV5.0 SP1 to SV5.0 SP2 Control Builder Compatibility Issues

| Condition | Issue | Solution |
|--|---|--|
| While upgrading, some project constants are removed from BasicLib (cEnable.*). | <ul style="list-style-type: none"> • If the removed project constants are used in the code, there are compilation errors after upgrading the project. • If the value of the project constants was changed in the project in 5.0 SP1, it leads to changed behavior of the code after the upgrade. This is because these project constants are used as default values for the parameters. | <ol style="list-style-type: none"> 1. Change to the correct literal value in the places where the project constants are used. 2. Check if the value of the removed project constant was changed in the project in 5.0 SP1. This can be done by opening the Project Constants dialog after the upgrade and checking if the project constants are listed. <p>In that case, the parameters previously using the project constant as a default value must be manually connected to a literal with the corresponding value.</p> |

SV4.x to SV5.0 SP2 Compatibility Issues

Table 11 lists and describes Control Builder compatibility issues, and the suggested solutions, when upgrading from SV4.x to SV5.0 SP2.

Table 11. SV 4.x to SV5.0 SP2 Control Builder Compatibility Issues

| Condition | Issue | Description / Solution |
|--|---|---|
| Mismatch in Application Change Analysis. | <p>Change Analysis Mismatches are shown for objects in the Standard Libraries during the first download after the upgrade.</p> <p>The possible mismatches are:</p> <p>Mismatch: Variable has changed data type.</p> <p>Mismatch: Variable not found.</p> <p>Mismatch: Control Module not found.</p> | <p>The mismatches only reflect the internal changes in the Standard Libraries; no cold retain values are lost.</p> <p>Click Next Mismatch.</p> |
| No Time Sync warning in the controller. | Some controllers get a No time sync warning in the controller log after the upgrade to 5.0 SP2. | Set the CS Protocol Type for the controllers, which are not intended to be synchronized, to No Synch . |

Table 11. SV 4.x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|---|---|--|
| Accessing Local Variables in Function Blocks. | <p>In 4.x, it is possible to access local variables in Function Blocks from surrounding code.</p> <p>In 5.0 SP2, which conforms to the IEC 61131-3 standard, local variables are accessible only within the containing software element. A compilation error is shown if local variables are accessed from the outside.</p> | Correct any such warnings before performing the download. Local variables used in this way need to be changed to parameters. |
| Code sorting loops treated as errors. | <p>By default, code sorting loops in applications are considered as errors in 5.0 SP2.</p> <p>It is not possible to compile and download an application if it contains code sorting loops.</p> | <p>Correct the sorting loops. Refer to Interpret and Correct Code Loop Errors in the manual <i>Industrial IT, 800xA - Control and I/O, Application Programming, Introduction and Design (3BSE043723Rxxxx)</i>.</p> <p>Another alternative is to change the compiler switch for Code Sorting Loops:</p> <ol style="list-style-type: none"> 1. Mark the project in Project Explorer. 2. Right-click and select Settings > Compiler Switch. 3. Set the global Loops in Control Modules switch to Warning. |

Table 11. SV 4.x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|--|--|---|
| <p>Large integer literal values as initial values.</p> | <p>In 4.x, illegal (too large) integer literal values are accepted as initial values. In 5.0 SP2, the Control Builder performs more stringent compiler checks on initial values for variables, and throws Compiler Error 1040.</p> | <p>Correct the compiler error (for example, by entering a smaller, legal value on the literal).</p> |
| <p>Self-defined serial communication using the Serial Communication Library.</p> | <p>The Serial Communication Library has undergone a major internal redesign.</p> | <p>Use of firmware functions Firmware functions used for handling serial communication are not supported in 5.0 SP2. This means that user-built libraries, which uses these firmware functions, can no longer be used. The firmware functions that are not supported in 5.0 SP2 are:</p> <ul style="list-style-type: none"> • OpenDevice • UpdateDeviceSetup • SetDeviceClearRead • CloseDevice • ReadStringDevice • ReadLineDevice • WriteStringDevice |

Table 11. SV 4.x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|--|--|--|
| Self defined serial communication using the Serial Communication Library. (continued) | The Serial Communication Library has undergone a major internal redesign that in some cases may lead to compatibility issues. (continued) | <p>Clear Buffer</p> <p>The Clear Buffer action is performed when the <i>listen</i> operation of the <i>SerialListen</i> function block is performed.</p> <p>SerialWriteWait</p> <ul style="list-style-type: none"> • If the <i>SerialWriteWait</i> function block is triggered when the time-out is elapsed, the read/listen buffer is cleared in between the retry attempts of the <i>write</i> operation. • In 4.x, if the function block is triggered after the application stopping phase is entered, no status is shown. The function block automatically triggers the write operation after the application change is completed. <p>In 5.0 SP2, if the function block is triggered after the application stopping phase is entered, the function block propagates the status code <i>-15</i> via the Status parameter.</p> <p>If the power fails, and a function block is in a pending state, the Status <i>-5331</i> is derived from it after the controller is powered up again.</p> |

Table 11. SV 4.x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|--|--|--|
| Self defined serial communication using the Serial Communication Library. (continued) | The Serial Communication Library has undergone a major internal redesign that in some cases may lead to compatibility issues. (continued) | <p>Printing a string longer than 140 characters:</p> <p>In 5.0 SP2, the behavior of the <i>SerialWrite</i> operation is not synchronous. The same function block cannot be called every time to print a string longer than 140 characters.</p> <p>To print a string longer than the maximum length of 140 characters, call subsequent function blocks in order:</p> <pre> Write1(Req := TRUE, Id := Id, EndChar := EndChar_Write1, Done => Done_Write1, Error => Error_Write1, Status => Status_Write1, Sd := Sd_Write1); Write2(Req := TRUE, Id := Id, EndChar := EndChar_Write2, Done => Done_Write2, Error => Error_Write2, Status => Status_Write2, Sd := Sd_Write2); </pre> <p>The printing of the two strings Sd_Write1 and Sd_Write2 (by calling the two function blocks) is queued, and they are printed in a series.</p> |

Table 11. SV 4.x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution | | | | | | | | | | | | | | | | | | | | | | |
|--|---|---|-----------------|--|--------|-------|--------------|----------------|-----------|-------------|--------------|----------------|---------------|-----------------|------------|--------------|--------------|----------------|------------|--|--------|-------|--------------------|----------------------|
| <p>Compiler warnings occur if there is a risk for task collisions.</p> | <p>If there is a risk that the tasks can collide in the controller, a warning is displayed during compilation:</p> <p>Warning 9155: Controller_1:HW Task Normal and Fast may have colliding start times</p> | <p>Apply proper task offsets, so that the tasks does not collide.</p> | | | | | | | | | | | | | | | | | | | | | | |
| <p>SattBus on TCP/IP.</p> | <p>SattBus on TCP/IP does not work after the upgrade to 5.0 SP2.</p> | <p>In 4.x, the COMLI communication function blocks are used for SattBus on TCP/IP.</p> <p>In 5.0 SP2, a new library, SattBusCommLib, which includes a set of new function blocks, is used.</p> <p>Change the application to use function blocks and data types from SattBusCommLib as follows:</p> <table border="0" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2" style="text-align: center;">Function Blocks</th> </tr> <tr> <th style="text-align: left;">Before</th> <th style="text-align: left;">After</th> </tr> </thead> <tbody> <tr> <td>COMLIConnect</td> <td>ComliSBConnect</td> </tr> <tr> <td>COMLIRead</td> <td>ComliSBRead</td> </tr> <tr> <td>COMLIReadCyc</td> <td>ComliSBReadCyc</td> </tr> <tr> <td>COMLIReadPhys</td> <td>ComliSBReadPhys</td> </tr> <tr> <td>COMLIWrite</td> <td>ComliSBWrite</td> </tr> <tr> <td>COMLIWriteDT</td> <td>ComliSBWriteDT</td> </tr> </tbody> </table> <table border="0" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2" style="text-align: center;">Data Types</th> </tr> <tr> <th style="text-align: left;">Before</th> <th style="text-align: left;">After</th> </tr> </thead> <tbody> <tr> <td>Comm_Channel_COMLI</td> <td>Comm_Channel_ComliSB</td> </tr> </tbody> </table> | Function Blocks | | Before | After | COMLIConnect | ComliSBConnect | COMLIRead | ComliSBRead | COMLIReadCyc | ComliSBReadCyc | COMLIReadPhys | ComliSBReadPhys | COMLIWrite | ComliSBWrite | COMLIWriteDT | ComliSBWriteDT | Data Types | | Before | After | Comm_Channel_COMLI | Comm_Channel_ComliSB |
| Function Blocks | | | | | | | | | | | | | | | | | | | | | | | | |
| Before | After | | | | | | | | | | | | | | | | | | | | | | | |
| COMLIConnect | ComliSBConnect | | | | | | | | | | | | | | | | | | | | | | | |
| COMLIRead | ComliSBRead | | | | | | | | | | | | | | | | | | | | | | | |
| COMLIReadCyc | ComliSBReadCyc | | | | | | | | | | | | | | | | | | | | | | | |
| COMLIReadPhys | ComliSBReadPhys | | | | | | | | | | | | | | | | | | | | | | | |
| COMLIWrite | ComliSBWrite | | | | | | | | | | | | | | | | | | | | | | | |
| COMLIWriteDT | ComliSBWriteDT | | | | | | | | | | | | | | | | | | | | | | | |
| Data Types | | | | | | | | | | | | | | | | | | | | | | | | |
| Before | After | | | | | | | | | | | | | | | | | | | | | | | |
| Comm_Channel_COMLI | Comm_Channel_ComliSB | | | | | | | | | | | | | | | | | | | | | | | |

Table 11. SV 4.x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|--|--|--|
| <p>Some project constants are removed from BasicLib (cEnable.*).</p> | <p>1. If these project constants are used in code, there are compilation errors after upgrading the project. 2. If the value of the project constants was changed in the project in 4.x, it leads to changed behavior of the code after the upgrade. This is because these project constants are used as default values on parameters.</p> | <p>1. Change to the correct literal value in the places where the project constants are used. 2. Check if the value of the removed project constant was changed in the project in 4.x. This can be done by opening the Project Constants dialog after the upgrade and checking if the project constants are listed. In that case, the parameters previously using the project constant as a default value must be manually connected to a literal value.</p> |

SV3.2/x to SV5.0 SP2 Compatibility Issues

Table 12 lists and describes Control Builder compatibility issues, and the suggested solutions, when upgrading from SV3.2/x to SV5.0 SP2.

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues

| Condition | Issue | Description / Solution |
|--|---|--|
| Mismatch in Application Change Analysis. | <p>Change Analysis Mismatches are shown for objects in the Standard Libraries during the first download after the upgrade.</p> <p>The possible mismatches are:</p> <p>Mismatch: Variable has changed data type.</p> <p>Mismatch: Variable not found.</p> <p>Mismatch: Control Module not found.</p> | <p>The mismatches only reflects internal changes in the Standard Libraries; no cold retain values are lost.</p> <p>Click Next Mismatch.</p> |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|--|---|--|
| <p>S900 I/O parameter name changes.</p> | <p>S900 I/O parameter names are changed, which causes loss of configuration data when upgrading a project containing S900 I/O modules.</p> <p>Error messages are shown during upgrade of the project.</p> | <p>Before migrating a project, repeat the following steps for all *.con files in the project that contains S900 I/O modules.</p> <ol style="list-style-type: none"> 1. Locate: CBMconS900Conv.exe in the tools\ABB\CBMconS900Conv directory on the CD. 2. Open a command prompt window and change to the directory where the CBMconS900Conv.exe is present: ... \> CD Z:\tools\abb\CBMconS900Conv\ 3. Enter CBMconS900Conv [path to project folder]ControllerName.con. For example: CBMconS900Conv ... \...Projects\MyProj\MyController.con 4. Press Enter. The MyController.con file is renamed to MyController.bak, and a new converted MyController.con file is generated. A message that the file is successfully converted appears. 5. Repeat Step 3 and Step 4 for all *.con files that include S900 I/O modules. |
| <p>No Time Sync warning in controller.</p> | <p>Some controllers get a No time sync warning in the controller log after upgrade to 5.0 SP2.</p> | <p>Set the CS Protocol Type for controllers that are not intended to be synchronized, to No Synch.</p> |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|---|---|--|
| Parameter name change in ProcessObjectAE Function block. | <p>The Name parameter on the ProcessObjectAE Function Block in AlarmEventLib is changed in 5.0 SP2. The parameter is now called CondNameObjErr.</p> <p>This means that applications using the ProcessObjectAE cannot be downloaded after the upgrade. There are compilation errors for unknown parameter names.</p> | Applications using ProcessObjectAE Function Block must be changed to use the CondNameObjErr parameter. |
| Some project constants are removed from BasicLib (cEnable.*). | <ol style="list-style-type: none"> 1. If these project constants are used in the code, there will be compiler errors after upgrading the project. 2. If the value of the project constants was changed in the project in 3.2/x, it will lead to changed behavior of the code after the upgrade, because these project constants are used as default values on parameters. | <ol style="list-style-type: none"> 1. Change to the correct literal value in the places where the project constants are used. 2. Check if the value of the removed project constant was changed in the project in 3.2/x. This can be done by opening the Project Constants dialog after the upgrade, and checking if the project constants are listed. <p>In that case the parameters previously using the project constant as a default value must be manually connected to a literal with the corresponding value.</p> |
| Process Object Libraries. | The 5.0 SP2 version contains enhanced versions (2.2/x) of the ProcessObjBasicLib and ProcessObjExtLib libraries. These are not fully compatible with the previous versions. | The previous versions (1.0/x) are included for upgrade reasons. Do not use them for new applications. |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|------------------------|--|---|
| System Library change. | <p>The SystemLib is split into two different libraries:</p> <p>System: Includes firmware functions and data types that are defined within firmware. This library is not version handled.</p> <p>BasicLib: Includes the rest of the former SystemLib. This library uses version handling.</p> | Both libraries must always be inserted into the project. |
| | Control Builder displays the Version Mismatch dialog when the project is downloaded after upgrade to 5.0 SP 2. | <p>Perform the following so that the Cold Retain Values are not lost:</p> <ol style="list-style-type: none"> 1. If the Version Mismatch dialog indicates: SystemLib not found click Rename, and enter BasicLib. 2. The Version Mismatch dialog also indicates several basic data types (for example, Real10, Bool10) that are located in the System Library. Since the System Library is automatically connected to the applications in the project, it is not necessary to rename the library. Click Next mismatch. |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|-----------------------|--|--|
| AlarmEventLib change. | <p>In 5.0 SP2, the SimpleEventDetector (AlarmEventLib) is based on GUID, instead of the name.</p> <p>The Source Name of a SimpleEventDetector should be identical to the Object Name of its owner. In that case, the changed behavior is not noticeable.</p> <p>If the two names are not identical, the SimpleEventDetector is associated with another object after the upgrade.</p> | Alter the application so that the naming complies with the naming recommendations. |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|-------------------------------|--|---|
| Communication Library change. | <p>In 5.0 SP2, the <i>CommunicationLib</i> is split into the following smaller libraries:</p> <ul style="list-style-type: none"> • MMSCommLib • FFH1CommLib • SattBusCommLib • ModBusCommLib • COMLICommLib • S3964CommLib • ModemCommLib <p>They are fully compatible with the former <i>CommunicationLib</i>.</p> | <p>Perform the following steps when replacing the old library:</p> <ol style="list-style-type: none"> 1. Open the project in which the <i>CommunicationLib</i> is still connected to the applications/libraries. 2. Insert the required new communication libraries to the project (For example, <i>MMSCommLib</i>, if the MMS protocol is used). 3. Connect the inserted libraries to the applications/libraries in the project. 4. Disconnect <i>CommunicationLib</i> from the applications/libraries. All Function Blocks are automatically replaced. <p>Perform the following additional steps for each instance of <i>CCToFF</i>, <i>FFT0CC</i>, <i>CCToMMS</i>, or <i>MMSToCC</i> in the applications/libraries:</p> <ol style="list-style-type: none"> 1. Connect the FFH1CommLib (<i>CCToFF</i>, <i>FFT0CC</i>) or MMSCommLib (<i>CCToMMS</i>, <i>MMSToCC</i>) to the application. 2. Mark the instance in Project Explorer. 3. Right-click and select the option to replace the Type. 4. Disconnect the <i>CommunicationLib</i>. 5. Select the Type from the new library. 6. Save and close. |
| | Control Builder displays the Version Mismatch dialog while downloading the project after the upgrade to 5.0 SP2. | <p>If the Version Mismatch dialog indicates:</p> <p style="padding-left: 40px;"><i>CommunicationLib</i> not found</p> <p>click Rename, and enter the name of the relevant communication library.</p> |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|------------------------|---|---|
| Serial Library change. | SerialLib is renamed to SerialCommLib. | Perform the following steps: <ol style="list-style-type: none"> 1. Open the project where SerialLib is still connected to the applications/libraries. 2. Insert the SerialCommLib library to the project. 3. Connect the SerialCommLib to the applications/libraries in the project. 4. Disconnect SerialLib from the applications/libraries. All Function Blocks are automatically replaced. |
| Library dependencies. | The use of circular dependencies causes compilation errors. | Circular dependencies between libraries is not permitted in 5.0 SP2. |
| ReallO Status change. | The status component of the ReallO data type is changed from dInt to dWord. | In most cases, this change happens automatically. But, if the status component is referred to directly in the application code, a manual change is necessary. This also applies for the ReallO status used in Process Graphics (For example, if the status is used in a faceplate). |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|---|--|---|
| Behavior of user-defined Hardware Definition Files when the 1131 application variables of type dInt and dWord are written from the application to I/O-channels. | <p>In 3.2/x, the behavior of user-defined Hardware Definition Files is undefined when the 1131 variable has a very large value (a value larger than the maximum possible value that can be written on a specific hardware unit).</p> <p>The very large value is truncated. For example, a hardware unit receiving 8-bit data received the 8 lowest bits in the dInt or dWord.</p> <p>In 5.0 SP2, this behavior is changed.</p> | <p>The behavior is redefined as follows:</p> <ul style="list-style-type: none"> • The hardware unit is written with the largest possible value (depending on the hardware) if the 1131 variable has a very large positive value. • The hardware unit is written with the largest negative value if the 1131 variable has a very large negative value. |
| Locations of Metafiles. | <p>Metafiles are not relocated on safe media during the source code migration. The image selectors fail if the paths to the image files are specified relative to the project directory.</p> | <p>Copy the files so that the specified relative path points to the files, or change the specified path to explicitly point to the image files (for example:</p> <pre> . . . \ImageDirectory\ImageName.wmf). </pre> <p>Windows metafiles (wmf) or enhanced metafiles (emf) that are used by image selectors in Control Module Diagrams are locally stored on disk, and they need to be present on all Control Builder M nodes. Only those files with wmf and emf images included in Control Module diagrams are affected.</p> |
| CI840 upgrade. | CI840 must be upgraded to firmware Version 3.0/0 or later. | CI840 must be upgraded to firmware Version 3.0/0 or later. |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|--|--|--|
| DP820 Scaling. | In 5.0 SP2, DP820 frequency scaling is changed. DP820 shows the wrong frequency value if this change is not done. | The default setting in SB 2.1/2 was 0.0 to 1500000.0 Hz. In the new version, it should be -15.00000.0 to +1500000.0 Hz. |
| Internal MMS Communication between applications residing in the same controller. | The behavior of internal MMS communication between two applications residing in the same controller is changed. | The internal MMS communication is asynchronous in 5.0 SP2; all the data is not always received in the same scan. |
| Naming Conventions. | <p>There are more stringent naming rules in 800xA System Version 5.x than in previous system versions. Therefore, in Control Builder SV 5.x, the instance names for variables, function blocks, control modules, etc. must be unique within the object where the instance is declared.</p> <p>If the instance names are not unique, when upgrading a project from an older system version and then trying to download it, error messages appear during compilation. For example,</p> <pre>The variable name <name> is not unique</pre> | Change the duplicate instance names so that they are unique. |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|---|--|--|
| Accessing Local Variables in Function Blocks. | <p>In 3.2/x, it is possible to access local variables in Function Blocks from the surrounding code.</p> <p>In 5.0 SP2, which conforms to the IEC 61131-3 standard, local variables are accessible only within the containing software element.</p> <p>A compilation error is shown if local variables are accessed from the outside.</p> | Correct any such warnings before performing the download. Local variables used in this way need to be changed to parameters. |
| Code sorting loops treated as errors. | By default, the code sorting loops in applications are considered as errors in 5.0 SP2. It is not possible to compile and download an application if it contains code sorting loops. | <p>Correct the sorting loops. Refer to Interpret and Correct Code Loop Errors in the manual <i>Industrial IT, 800xA - Control and I/O, Application Programming, Introduction and Design</i> (3BSE043723Rxxxx).</p> <p>Another alternative is to change the compiler switch for Code Sorting Loops:</p> <ol style="list-style-type: none"> 1. Mark the project in Project Explorer. 2. Right-click and select Settings > Compiler Switch. 3. Set the global Loops in Control Modules switch to Warning. |
| Large integer literal values as initial values. | In 5.0 SP2, the Control Builder performs more stringent compiler checks on initial values for variables. It does not allow illegal (too large) integer literal values as initial values. | Correct the compile error (for example, by entering a smaller, legal value on the literal). |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution | | | | | | | | | | | | | | |
|--|--|--|--------|-------|--------------|----------------|-----------|-------------|--------------|----------------|---------------|-----------------|------------|--------------|--------------|----------------|
| SattBus on TCP/IP. | SattBus on TCP/IP does not work after the upgrade. | <p>In 3.2/x, the COMLI communication function blocks are used for SattBus on TCP/IP.</p> <p>In 5.0 SP2, a new library, SattBusCommLib, which includes a set of new function blocks, is used.</p> <p>Change the application to use function blocks and data types from SattBusCommLib as follows:</p> <p style="text-align: center;">Function Blocks</p> <table style="width: 100%; border: none;"> <tr> <td style="text-align: center;">Before</td> <td style="text-align: center;">After</td> </tr> <tr> <td>COMLIConnect</td> <td>ComliSBConnect</td> </tr> <tr> <td>COMLIRead</td> <td>ComliSBRead</td> </tr> <tr> <td>COMLIReadCyc</td> <td>ComliSBReadCyc</td> </tr> <tr> <td>COMLIReadPhys</td> <td>ComliSBReadPhys</td> </tr> <tr> <td>COMLIWrite</td> <td>ComliSBWrite</td> </tr> <tr> <td>COMLIWriteDT</td> <td>ComliSBWriteDT</td> </tr> </table> <p style="text-align: center;">Data Types</p> <p>Comm_Channel_COMLI is changed to Comm_Channel_ComliSB</p> | Before | After | COMLIConnect | ComliSBConnect | COMLIRead | ComliSBRead | COMLIReadCyc | ComliSBReadCyc | COMLIReadPhys | ComliSBReadPhys | COMLIWrite | ComliSBWrite | COMLIWriteDT | ComliSBWriteDT |
| Before | After | | | | | | | | | | | | | | | |
| COMLIConnect | ComliSBConnect | | | | | | | | | | | | | | | |
| COMLIRead | ComliSBRead | | | | | | | | | | | | | | | |
| COMLIReadCyc | ComliSBReadCyc | | | | | | | | | | | | | | | |
| COMLIReadPhys | ComliSBReadPhys | | | | | | | | | | | | | | | |
| COMLIWrite | ComliSBWrite | | | | | | | | | | | | | | | |
| COMLIWriteDT | ComliSBWriteDT | | | | | | | | | | | | | | | |
| Compiler warnings for task collisions. | <p>If there is a risk that tasks can collide in the controller, a warning is displayed during compilation:</p> <pre style="margin-left: 40px;">Warning 9155: Controller_1:HW Task Normal and Fast may have colliding start times</pre> | Apply proper task offsets so that the tasks can no longer collide. | | | | | | | | | | | | | | |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|--|--|---|
| <p>Self defined serial communication using the Serial Communication Library.</p> | <p>The Serial Communication Library has undergone a major internal redesign.</p> | <p>Use of firmware functions: Firmware functions used for serial communication handling are not supported in 5.0 SP2. This means that the user-built libraries, which used these firmware functions, can no longer be used. The firmware functions that are not supported in 5.0 SP2 are:</p> <ul style="list-style-type: none"> • OpenDevice • UpdateDeviceSetup • SetDeviceClearRead • CloseDevice • ReadStringDevice • ReadLineDevice • WriteStringDevice |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|---|---|---|
| Self defined serial communication using the Serial Communication Library. (continued) | The Serial Communication Library has undergone a major internal redesign. (continued) | <p>Clear Buffer</p> <p>The Clear Buffer action is performed when the <i>listen</i> operation of the <i>SerialListen</i> function block is performed.</p> <p>SerialWriteWait</p> <ul style="list-style-type: none"> • If the <i>SerialWriteWait</i> function block is triggered when the time-out is elapsed, the read/listen buffer is cleared in between the retry attempts of the write operation. • In 3.2/x, if the function block is triggered after the application stopping phase is entered, no status is shown. The function block automatically triggers the write operation after the application change is finished. <p>In 5.0 SP2, if the function block is triggered after the application stopping phase is entered, the function block propagates the status code <i>-15</i> via the Status parameter.</p> <p>If the power fails while a function block is in a pending state, the Status <i>-5331</i> is derived from it after the controller is powered up again.</p> |

Table 12. SV3.2/x to SV5.0 SP2 Control Builder Compatibility Issues (Continued)

| Condition | Issue | Description / Solution |
|--|--|---|
| <p>Self defined serial communication using the Serial Communication Library. (continued)</p> | <p>The Serial Communication Library has undergone a major internal redesign. (continued)</p> | <p>Printing a string longer than 140 characters:</p> <p>In 5.0 SP2, the behavior of the <i>SerialWrite</i> operation is not synchronous. The same function block cannot be called every time to print a string longer than 140 characters.</p> <p>To print a string longer than the maximum length of 140 characters, call the subsequent function blocks in order:</p> <pre>Write1(Req := TRUE, Id := Id, EndChar := EndChar_Write1, Done => Done_Write1, Error => Error_Write1, Status => Status_Write1, Sd := Sd_Write1);</pre> <pre>Write2(Req := TRUE, Id := Id, EndChar := EndChar_Write2, Done => Done_Write2, Error => Error_Write2, Status => Status_Write2, Sd := Sd_Write2);</pre> <p>The printing of the two strings Sd_Write1 and Sd_Write2 (by calling the two function blocks) is queued, and they are printed in a series.</p> |

Appendix D Communication Cables

Serial communication between Control Builder and the PLC is performed by using the TK212A cable.

Connect the DB9 Female connector to a Control Builder PC COM port, thus the RJ45 (8P8C) plug to the PLC COM4 port. The [Figure 91](#) illustrates the TK212A pin-out configuration.

Connecting Control Builder PC to a PLC

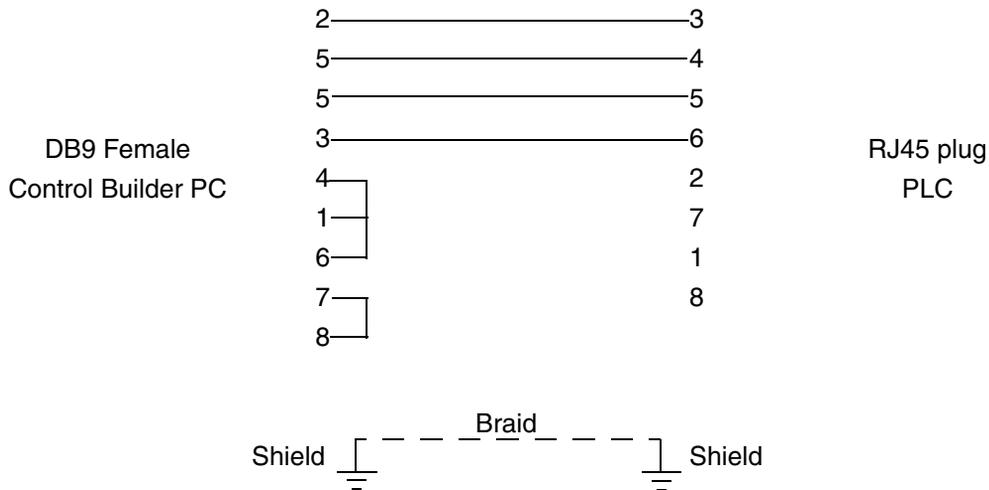


Figure 91. Cable TK212A Pinout configuration.

Appendix E Programming Languages

Compact Control Builder provides five different programming languages according to IEC 61131-3. These are Function Block Diagram (FBD), Structured Text (ST), Instruction List (IL), Ladder Diagram (LD) and Sequential Function Chart (SFC). The specific rules and syntax of the programming languages will not be discussed in detail in this manual. For more information, refer to the Control Builder Online Help.



SFC Viewer is not supported by Compact Control Builder AC 800M.

General

The IEC 61131-3 standard defines five of the most commonly used programming languages on the market. Depending on previous experience, programmers often have their own personal preference for a certain language. All the languages have advantages and disadvantages, and no single one of them is suitable for all control tasks. We start with three basic statements and then proceed to some important characteristics of each language.

- In small applications with relatively few logical conditions, the demand for good structure and re-use of code is not as great as in larger systems.
- ST and IL are textual languages, while FBD, LD, and SFC are based on graphical metaphors.
- LD and IL are not as powerful as ST or FBD.



The definition of function block is allowed in all five languages, not only in FBD. A function block is a method of encapsulating the code in a “black box” with inputs and outputs.

All instructions and functions viewed in this section are explained in detail in Control Builder online help.

For more information on the different languages, refer to the manual *IEC 61131 Control Languages, Introduction*.

Some important characteristics of the languages are listed in [Table 13](#).

Table 13. Compact Control Builder programming languages.

| Language | Function |
|---------------------------------|---|
| Function Block Diagram (FBD) | A graphical language for depicting signal and data flows through function blocks and re-usable software elements. Function blocks and variables are interconnected graphically, which makes the resulting control diagrams easy to read. |
| Structured Text (ST) | A high-level programming language. ST is highly structured and has a comprehensive range of constructs for assignments, function/function block calls, expressions, conditional statements, iterations, etc. It is easy to write advanced, compact, but clear ST code, due to its logical and structured layout. |
| Instruction List (IL) | A traditional PLC language. It has a structure similar to simple machine assembler code. |
| Ladder Diagram (LD) | Ladder diagram (LD) is a graphical language based on relay ladder logic. |
| Sequential Function Chart (SFC) | Sequential function chart (SFC) is a graphical language for depicting the sequential behavior of a control program. |

Appendix F Glossary

The following is a list of terms associated with Compact Control Builder. The list contains some terms and abbreviations that are unique to ABB or have a usage or definition that is different from standard industry usage.

| Term/Acronym | Description |
|---------------------------------------|---|
| Application | The application contains the program code to be compiled and downloaded for execution in the PLC. Applications are displayed in the Project Explorer. |
| Control Builder | A programming tool with a compiler for control software. Control Builder is accessed through the Project Explorer interface. |
| Control Module (Type) | A program unit that supports object-oriented data flow programming. Control modules offer free-layout graphical programming, code sorting and static parameter connections. Control module instances are created from control module types. |
| Firmware | The system software in the hardware. |
| Hardware Description | The tree structure in the Project Explorer, that defines the hardware's physical layout. |
| IEC 61131-3 | A standard for control system languages defined by the IEC. |
| Industrial ^{IT} | ABB's vision for enterprise automation. |
| Industrial ^{IT} 800xA System | A computer system that implements the Industrial IT vision. |
| Interaction Window | A graphical interface used by the programmer to interact with an object. Available for many library types. |

| Term/Acronym | Description |
|------------------|---|
| MMS | Manufacturing Message Specification, a standard for messages used in industrial communication. This is the application layer used within MAP (Manufacturing Automation Protocol), a specification for open communication based on the OSI model. |
| OPC | OLE for Processing Control. |
| OPC/DA | An application programming interface defined by the standardization group OPC Foundation. The standard defines how to access large amounts of real-time data between applications. The OPC standard interface is used between automation/control applications, field systems/devices and business/office application. |
| Process Object | A process concept/equipment such as valve, motor, conveyor or tank. |
| Project Explorer | The Control Builder interface. Used to create, navigate and configure libraries, applications and hardware. All objects such as data types, functions and function block types can be selected and displayed in an editor. The software and hardware is configured in the Project Explorer. |
| RNRP | Redundant Network Routing Protocol. |

| Term/Acronym | Description |
|---------------------|--|
| Security | <p>Security controls a user's authority to perform different operations on (Aspect) Objects, depending on several parameters:</p> <ul style="list-style-type: none">• The user's credentials, as provided by Windows• The node where the user is logged in. This makes it possible to give users different authority depending on where they are located, for example close to the process equipment, in a control room, or at home accessing the system through Internet.• The task that the user wants to perform. |
| Type | <p>A type solution that is defined in a library or locally, in an application. A type is used to create instances, which inherit the properties of the type.</p> |

INDEX**A**

AC 800M
 cable configuration 171
 SB 2.1/2 to SV 4.0 137
 Product upgrade
 SB 2.1/2 to SV 4.0 138
 AC800M 32
 addresses
 convert 128
 Alarm and Event Library 15
 analysis 111
 changes at download 112
 test mode 61
 application 31
 Applications 35

B

Basic Library 15, 45

C

Cable Configuration
 AC 800M 171
 change analysis 112
 Check 39
 Check icon 37
 class A
 IP address 127
 class B
 IP address 127
 class C
 IP address 127
 client
 node 129
 client/server networks 129
 code pane 46
 Cold Retain 139
 Communication Library 15
 compilation 112
 configuration

MMS 131
 Connected Libraries 36
 Contents Topic 21
 Control builder M
 Compatibility issues 156
 Copying files to safe media
 SB 2.1/2 to SV 4.0 141
 Control Module Types 37
 control modules 17
 Control Network 129
 Controllers 35
 controllers
 select at download 114
 convert
 addresses 128
 Copying files to safe media
 Control builder M
 SB 2.1/2 to SV 4.0 141
 OPC server for AC 800M
 SB 2.1/2 to SV 4.0 140
 counters 49
 cross-over 85

D

Data Types 37
 declaration pane 46
 Description 39
 Difference Report 61, 89
 download
 new project 114
 select controller(s) 114
 to selected controller(s) 114
 Download and Go Online 113
 Downloading
 via Ethernet 86

E

editor
 programs 46

EmptyProject 32
Enable Ethernet channel 123
Ethernet
 download via 86
 ports 87
 set IP address 87
Ethernet cable 85

F

FBD 173
File Location
 select 97
File server 99
Firmware
 upgrade via serial line 79
Firmware Version 81
Forced 90
Function Block Types 37

H

Heap 94
Heap Utilization 146
host ID
 IP address 128
HostID 127

I

I/O Address 78
I/O channels 74
Icon library 45
IL 173
implicit IP addressing 119
Index 21
index
 online help 21
Industrial IT licenses 137
IP address 82, 84
 class A 127
 class B 127
 class C 127

host ID 128
IPConfig 82

K

Keyword Search 21

L

Language
 select 96
LD 173
Libraries 35
 AlarmEventLib 15
 BasicLib 15, 45
 CommunicationLib 15
 Control Library 15
local flag 128

M

manuals 20
Memory 94
Message 39
message pane 46, 60
mismatch 146
MMS
 configuration 131
ModuleBus 72
multi-user configuration 19

N

NetID 127
network area 127 to 128
network ID 128
node
 client 129
 server 129
node number 127
RNRP 128

O

online analysis 89 to 90
 Online Documentation 20
 Online Help 20
 online help
 index 21
 text search 22
 Online Manuals 20
 online mode 89 to 90
 OPC panel 29
 OPC Server 23
 OPC server for AC 800M
 Copying files to safe media
 SB 2.1/2 to SV 4.0 140

P

parameters
 RNRP 127
 path number
 RNRP 128
 Permission 99
 permission Full Control 100
 PLC Control Builder 23
 PLC control builder version 79
 PLC firmware 79
 ports
 Ethernet 87
 IP address 87
 program 31
 program editor 46
 project 31
 project documentation 20
 Project Explorer 14
 Project folder 19, 97

R

real-time communication 129
 refresh 33
 Restoring files from safe media
 Control builder M, compatibility issues 156

RNRP

node number 128
 parameters 127
 path number 128
 RNRP Service 142
 RNRP Setup Wizard 120
 RNRP-icon 120

S

SB 2.1/2 to SV 4.0
 AC 800M considerations
 License upgrade 137
 Product upgrade 138
 Copying files to safe media
 Control builder M 141
 OPC server for AC 800M 140
 search
 online help 22
 Selecting
 file location 97
 language 96
 server
 node 129
 Service Account 106 to 107
 session log file 28
 Settings tab 87
 Setup Wizard 94
 SFC 173
 single-user configuration 23
 SoftController 32
 software model 69
 ST 173
 standard libraries 15
 straight-through 85
 structuring code 51
 subnetwork ID 128
 System folder 32

T

Tasks 38

templates 32
test mode 60 to 61
 analysis 61
Timers 49
TK212A cable 79, 171

U

UNC path 101, 105
Upgrade firmware via serial line 79
upgrading projects 133

V

variable conditions 63
version and online analysis 111
versions 111



3BSE041584R5022
Copyright © 2003-2008 by ABB. All Rights Reserved
® Registered Trademark of ABB.
™ Trademark of ABB.

<http://www.abb.com>

Automation Technology Products
Västerås, Sweden
www.abb.com/controlsystems

Automation Technology Products
Wickliffe, Ohio, USA
www.abb.com/controlsystems