Application on Open-Loop Control & Closed-Loop Control

PC-Based Automation with SIMATIC WinAC

Connection of Databases via open Interfaces Using OPC-XML, Programmed in C#.net
Warranty, Liability and Support

We accept no liability for information contained in this document.

We do not accept liability, whatever the legal basis, for any damages arising from the use of examples, notes, programs, configuration and performance data etc. described in this document, except where we are obliged to by the German Product Liability Act or in cases of willful damage or gross negligence, injury to life, body or health, breach of guarantee for the condition of products or items assumed by us, fraudulent concealment of a defect or breach of a substantial contractual obligation. However, claims arising from a breach of a condition which goes to the root of the contract shall be limited to the foreseeable damage which is intrinsic to the contract, unless caused by intent or gross negligence or based on mandatory liability for injury of life, body or health. The above conditions are not meant to change the burden of proof to the detriment of the user.

The application examples do not purport to cover all details or variations in equipment, nor do they provide for every possible contingency. They are not customer-specific solutions. They are only intended to provide support for typical tasks. You are responsible for ensuring that the described products are used correctly. These application examples do not relieve you of the responsibility to use sound practices in application, installation, operation and maintenance. By using this application example you accept that Siemens is not liable for any damages except for those specified in the above liability clause. We reserve the right to make changes in these application examples at any time without prior notice. If there are any deviations between the recommendations provided in this application example and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

Copyright © 2005 Siemens A&D. Any form of duplication or distribution of these application examples or excerpts hereof is prohibited without the expressed consent of Siemens A&D.

For questions about this document please use the following e-mail address:

mailto:csweb@ad.siemens.de
Foreword

Objective of the application

Eight examples were developed to provide a quick lead-in to PC-based automation with SIMATIC WinAC. They consist of the sample code and an extensive documentation. Using these examples, the user can familiarize with the individual topics on a task-specific basis.

Main contents of this application

This application deals with the following key elements:

- Linking databases to WinAC RTX
- Basics of Microsoft .net
- Database accesses via the ADO.net interface
- Basics of Microsoft ACCESS and MySQL

Delimitation

This application does not include a detailed description of

- the setup of databases
- the basics of OPC XML
- programming in C#

Basic knowledge of the topics listed above is required; for detailed information, please refer to other applications of this application series. The bibliographic references includes references.

The individual examples

To enable optimum use of PC-based automation, we have developed one example from the “classic” PLC world and one from the “open” PC world for each of the four typical automation tasks (controlling, communication, visualization, technology).

All eight examples with their allocation to the respective automation tasks are shown in the figure below. This example, which deals with "Data Exchange via OPC XML", is displayed with a red margin.
**Basis of the examples**

All examples are based on a virtual "mixing process". Using this "mixing process", the different tasks and automation components from the product range of PC-based automation are applied.

**System picture**

The following figure shows the system picture of the "mixing process". The red margin indicates the components described in this example.
Basic solution approach of this application

The application requires different hardware and software components. Some of these components are included in the delivery of this application, others are provided by you.

This documentation describes the individual components and their interaction.
Foreword

Structure of the document

The documentation of this application is divided into the following main parts.

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Part A1 provides you with a general overview of the contents. You are informed on the components used (standard hardware and software components and the additionally developed software). The displayed basic function data show the performance capability of this application.</td>
<td>You can skip this part if you want to test the application first using the step-by-step instructions.</td>
</tr>
<tr>
<td>A2</td>
<td>Part A2 provides a detailed description of the function processes of the hardware and software components involved. It is only required to read this part if you are interested in the detailed process and the interaction of the solution components.</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Part B takes you step by step through configuration and startup of the application.</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Part C is of interest if you want to expand or adapt the software to your system.</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Part D “Appendix” includes further information, e.g. bibliographic references and a feedback questionnaire for your comments on this document.</td>
<td></td>
</tr>
</tbody>
</table>

Reference to Automation and Drives Service & Support

This entry is from the Internet application portal of Automation and Drives Service & Support. Clicking the link below directly displays the download page of this document.

http://support.automation.siemens.com/WW/view/de/21576581
# Table of Contents

Table of Contents ......................................................................................................... 7

1  Automation Task ........................................................................................................ 10
2  Automation Solution .................................................................................................. 12
2.1 Overview of the overall solution ............................................................................. 16
2.2 Description of the functionalities of the application .............................................. 19
2.3 Alternative solutions .............................................................................................. 21
2.4 Required components ............................................................................................. 23
3  Performance Data ..................................................................................................... 27
4  Function Mechanisms ............................................................................................... 29
4.1 Basics: PC-based control with SIMATIC .............................................................. 29
4.2 Web services, IIS and OPC XML-DA ..................................................................... 29
4.3 Microsoft .NET platform ....................................................................................... 29
4.4 Basics of the C# programming language ............................................................... 30
4.5 Basics of the ADO.NET interface .......................................................................... 31
4.6 Basics of ACCESS ................................................................................................ 35
4.7 Basics of MySQL .................................................................................................. 36
5  Installation of Hardware and Software ........................................................................ 38
5.1 Preparatory installation .......................................................................................... 38
5.2 Hardware installation ............................................................................................ 39
5.3 Installation of WinAC RTX .................................................................................... 41
5.4 Installation of the MySQL database server ............................................................ 42
5.5 Installation and setup of the Internet Information Services ................................. 44
5.6 Installation of the STEP 7 project .......................................................................... 46
5.7 Installation of the Windows Bridge Client .............................................................. 47
5.8 Configuration and programming of WinAC ........................................................... 47
5.9 Configuration of the MySQL database ................................................................. 48
5.10 Configuration and programming of the ACCESS database ............................... 48
6  Operating the Application ......................................................................................... 49
6.1 Operating the MySQL database ............................................................................. 49
6.2 Operating the Windows application (Bridge Client) ......................... 52
6.2.1 Automatic mode ........................................................................... 56
6.2.2 Manual mode ................................................................................ 59
6.2.3 VIEW (visualization) ................................................................... 62
6.3 Operating the S7 control program ................................................... 63
7 Explanation of the Codes of the Individual Elements ....................... 68
7.1 Overview of the individual software components ............................ 68
7.2 Recipe control (S7 control program) .............................................. 68
7.2.1 Program structures ....................................................................... 68
7.2.2 Program flow ................................................................................ 69
7.2.3 Overview of function blocks and organization blocks ................. 71
7.2.4 Overview of the data blocks ......................................................... 76
7.2.5 Description of selected code excerpts ......................................... 78
7.3 Bridge Client (C# program) ............................................................. 80
7.3.1 Principles of operation of the client ............................................. 80
7.3.2 OPC-XML access (C# Windows program) ................................ 83
7.3.3 Database accesses (C# Windows program) ................................ 85
7.3.4 Visualization (C# Windows program) ......................................... 92
7.4 Database structure (ACCESS / MySQL) ......................................... 93
7.4.1 Tables .......................................................................................... 93
8 Adapting the Programs ................................................................. 95
8.1 Changes in the S7 program ............................................................. 95
8.2 Adapting the C# application .......................................................... 96
8.3 Creating a strong-typed DataSet .................................................... 97
9 Bibliographic References ............................................................... 100
Part A1: Application Description

Overview

Contents of Part A1

Part A1 provides you with a general overview of the contents. You are informed on the components used (standard hardware and software components and the additionally developed software).

The displayed basic function data show the performance capability of this application.

Objectives of Part A1

Part A1 of this document provides the reader with the following:

- Explanation of the automation problem
- Illustration of a possible solution
- Illustration of the performance capability of the overall application.

Topics

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Automation Task</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Automation Solution</td>
<td>12</td>
</tr>
<tr>
<td>2.1</td>
<td>Overview of the overall solution</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Description of the functionalities of the application</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Alternative solutions</td>
<td>21</td>
</tr>
<tr>
<td>2.4</td>
<td>Required components</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Performance Data</td>
<td>27</td>
</tr>
</tbody>
</table>
1 Automation Task

Requirement

Industrial PCs, which are used for visualization, are frequently used in the field of industrial automation. A controller which controls the system exists simultaneously. A frequent problem is that data of a database are to be transferred to the controller and vice versa.

General task

Data are to be exchanged between a database and a SIMATIC controller via OPC XML server. SIMATIC WinAC RTX is to be used as controller.

Context of this automation task

The application examples on the topic, to which this document also belongs, are based on the mixing and bottling plant for refreshments displayed in the above figure. The mixing process is controlled by the SIMATIC WinAC RTX software controller.

General technological task

This application shows how SIMATIC WinAC RTX can interact with databases via the SIMATIC OPC-XML server. The standard components of Microsoft .NET Framework are used, which is available free of charge.

To make the example more specific, we show you the recipe-based control of the mixing process in WinAC; the recipes are read from the database and additionally quality data are written back to the database.

However, the example focuses on linking the WinAC RTX controller to a database using OPC-XML server.

Specific technological task

The MS ACCESS (MySQL) database is to communicate with WinAC RTX. A Windows application, which is also used for visualization, is to be integrated as Bridge client between database and controller. The data are to be exchanged via this application.

Solution requirements

- The application is to provide the following functionality:
  1. For the moment, recipe processing is in idle state
  2. Start of the recipe with preselected recipe
  3. Controller requests data at the Windows client
  4. Processing of the recipe using the recipe (dynamic)
  5. After completion of the recipe, the required time of the recipe and the protocol data of the individual steps are stored in the database.
- Functionality applicable to large quantity frameworks.
The recipe program in the controller is to be fully operable from the client application.

All functions also have to be operable from the S7 program in WinAC via a variable table.

The connected I/O is simulated by a simulation block.

It has to be possible to transfer different dynamic recipes to the controller.

After each completed recipe process, the WinAC quality data have to be transferred to the database.

Standardized communication mechanisms are to be used.
2 Automation Solution

Introduction

This chapter provides you with specific information on how this application solves the automation task described in Chapter 1. It illustrates what the application can perform, which submodules it contains and how they work. The functions are deliberately described in universally applicable terms. Part A2 of this documentation provides in-depth information which you need only if you are interested in background information, the detailed process and the interaction of the individual solution components.

Contents of this chapter

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Automation Solution</td>
<td>12</td>
</tr>
<tr>
<td>2.1</td>
<td>Overview of the overall solution</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Description of the functionalities of the application</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Alternative solutions</td>
<td>21</td>
</tr>
<tr>
<td>2.4</td>
<td>Required components</td>
<td>23</td>
</tr>
</tbody>
</table>

Possible solutions

A solution with WinAC RTX in connection with the OPC server is predestined to solve such tasks. WinAC RTX provides the PC with the control functionality. The OPC server offers the option to exchange data between Windows applications and the controller. This works on a local as well as on a distributed basis with other computers.

Example application

The example for such an automation solution presented in this document is the linking of an ACCESS or MySQL database to the WinAC RTX controller.

Benefit

The completely prepared and executable application software provides the following benefit:

- Options for creating a dynamic recipe control in WinAC are shown
- Basics required to enable an individual data exchange between databases and WinAC using OPC server are provided
  - Programming a database link using ADO.NET
  - Programming an OPC client (brief)
This application

This document describes an application example suitable to establish a connection to an MS ACCESS (or MySQL) database with the aid of a self-created Windows application, as well as for the transfer of data as recipe information to the WinAC RTX controller via the OPC-XML server. In turn, the Windows application receives quality data from the controller, which are then stored in the database.

Note

Testing the application does not require a real mixing process, since this process is simulated by a function block in the controller.

Architecture of this application

The figure below illustrates the structure of the overall application.

The OPC client (in the following also referred to as Bridge Client) is the central component for the connection between WinAC RTX and database. It sends and receives data to and from the controller and the database.

Installation structure 1 (only local components)

The illustration below shows the basic technological structure of the solution. The data exchange between WinAC RTX and database is effected via the Bridge Client Windows application.
For a detailed description of the data exchange between the individual components, please refer to Chapter 2.2 Description of the functionalities of the application.

Installation structure 2 (database on remote server)

This application also enables the connection to a database (ACCESS / MySQL) on a remote server. The connection to the server is established with the corresponding settings in the client.

Installation structure 3 (database and client on remote computer)

A third option for the hardware configuration is to install the Bridge Client on a remote computer. The connections can be freely selected by setting the client accordingly.
Note: The three options described above are only a selection of possible hardware configurations. In the following, only the first two options are explained.

Quantity framework of the example

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Performance data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load memory requirement</td>
<td>Approx. 5400 bytes</td>
</tr>
<tr>
<td>Main memory requirement</td>
<td>Approx. 4000 bytes</td>
</tr>
<tr>
<td>Cyclic, asynchronous reading</td>
<td>7 words</td>
</tr>
<tr>
<td>User-controlled, synchronous writing</td>
<td>7 words for recipe workstep, i.e. a total of 14 ... 140 DBW + approx. 3 bytes for control codes</td>
</tr>
</tbody>
</table>

Note: The size of the recipe block limits the dynamic recipes to 20 individual steps. The recipe size can be expanded very easily (see Chapter 8 Adapting the Programs).

Customer benefit from this automation solution

Compared to typical recipe control systems operated without a PC, this solutions offers the following advantages:

- The solution includes an easy-to-use visualization function
- All quality and error data of the process are archived in the database and are then available for error analysis.
• All classic control functionalities can be realized with WinAC as usual and communicate with other Windows applications via the OPC-XML client connection (locally as well as with databases via networks).

• The individually implemented client application enables to achieve an efficient data exchange with standard databases. Database and client application can be installed on different computers (connected via Ethernet). (If required, each of the three sub-applications WinAC, Bridge Client and database can be installed on a separate computer).

• The dynamic structure of the C# program enables the use of the code elements in new projects.

Teaching material of this application

After studying this application example, you will be familiar with the following:

• How to program an OPC-XML access to data of WinAC RTX from a C# application.

• How the ADO.Net interface works.

• How to connect an ACCESS/MySQL database via ADO.NET.

2.1 Overview of the overall solution

Display of the components involved

The figure below shows the hardware configuration of the example application, as well as the standard and user software components involved.

Please note that the application is operated without I/O devices. This is realized with a simulation block in the S7 program which simulates the I/O. A real connection of I/O requires changing the S7 program. (See Chapter 8.1 Changes in the S7 program)
**Explanation of the components**

The WinAC station contains the PLC software, the OPC server and provides the data for the controller.

**Basic functionality of the solution**

The Bridge Client reads the recipe data from one of the two databases and writes it to a data block of the controller (green communication lines). In addition, quality data are transferred back to the database (red communication lines). If required, the controller transmits visualization data (blue communication line). The illustration below shows an overview of the software solution.
Overview of the software solution

Figure 2-6
2.2 Description of the functionalities of the application

Solution elements of the application

The application consists of several solution elements which are connected on a very variable and open basis:

Table 2-2

<table>
<thead>
<tr>
<th>No.</th>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WinAC RTX</td>
<td>A controller with real-time capability. It performs the actual control of the mixing process.</td>
</tr>
<tr>
<td>2</td>
<td>SIMATIC NET OPC server</td>
<td>Establishes a connection between controller and Internet Information Services.</td>
</tr>
<tr>
<td>3</td>
<td>Internet Information Services (IIS)</td>
<td>Establish a SOAP-based web connection for local clients as well as for remote clients to the OPC server.</td>
</tr>
<tr>
<td>4</td>
<td>OPC client</td>
<td>Retrieves the data from the database with the aid of ADO.NET mechanisms and forwards it to the OPC server. In addition, the client provides an easy-to-use visualization function for operator control and monitoring of the S7 program.</td>
</tr>
<tr>
<td>5</td>
<td>ADO.NET</td>
<td>Database link architecture for .NET Framework.</td>
</tr>
</tbody>
</table>
Database Link to WinAC via OPC-XML

6 MySQL data provider Makes the data of a MySQL database available to the ADO.NET architecture (DataSet etc.).

7 MS Jet OLE DB data provider Makes the data of a Jet-compatible database (e.g.: MS Access) available to the ADO.NET architecture (DataSet etc.).

8 MS ACCESS database Database for storage of the recipe and quality data.

9 MySQL database Database for storage of the recipe and quality data.

10 SOAP SOAP is a protocol standard based on XML for distributed communication in computer networks.

Sequence of selected core functionalities

The tables below describe the actions performed by the application to realize selected core functionalities:

Connecting the OPC client to the server

Table 2-3

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initializing the web service objects</td>
<td>An instance of the web reference is generated and filled with initialization data (timeout, URL, etc.).</td>
</tr>
<tr>
<td>2</td>
<td>Requesting a WebResponse by the client</td>
<td>This mechanism is used to check whether the OPC_XML server is installed and available.</td>
</tr>
<tr>
<td>3</td>
<td>Calling the operating status of the server.</td>
<td>Further check by requesting server status information.</td>
</tr>
<tr>
<td>4</td>
<td>If required: Establishing the connection</td>
<td>If errors occur, corresponding messages are displayed and no connection is established.</td>
</tr>
</tbody>
</table>

Connecting the client to the database

Table 2-4

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Opening connection to the database.</td>
<td>Using the DataAdapter (connection), the Access (MySQL) database is opened for connections.</td>
</tr>
<tr>
<td>2</td>
<td>Saving the data in the DataSet</td>
<td>The fill method of the data adapter fills a DataSet with the data from the database.</td>
</tr>
<tr>
<td>3</td>
<td>Closing the connection to the database</td>
<td>After processing all requests, the DataAdapter automatically closes the connection to the database. The data are stored in the DataSet and can be edited.</td>
</tr>
</tbody>
</table>
Data exchange between database and S7 controller

Table 2-5

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start of the controller</td>
<td>Via the &quot;Start&quot; bit in the DB_Command_Glob data block, the control is started either by the controller or the client.</td>
</tr>
<tr>
<td>2</td>
<td>The controller sends a request for recipe data.</td>
<td>The controller sets the &quot;DataRequest&quot; bit in the DB_Command_Glob data block.</td>
</tr>
<tr>
<td>3</td>
<td>The OPC client acknowledges the request and sends the recipe data to the DB_Recipe data block.</td>
<td>In the DB_Command_Glob data block of the controller, the client resets the &quot;DataRequest&quot; bit and enables the &quot;DataRequestAk&quot; bit.</td>
</tr>
<tr>
<td>4</td>
<td>The OPC client sends a message acknowledging the sending of data.</td>
<td>The client sends the recipe data to the DB_Recipe data block of the controller using a synchronous write command. Subsequently, the &quot;DataSend&quot; bit is set in DB_Command_Glob of the controller.</td>
</tr>
<tr>
<td>5</td>
<td>The controller sends a data reception message and starts the recipe program.</td>
<td>WinAC resets the &quot;DataSend&quot; bit and activates the &quot;DataSendAk&quot; bit.</td>
</tr>
</tbody>
</table>

Advantages of this solution

- The database link can be freely selected, to the greatest possible extent independently of firewalls.
- Individual and flexible, self-created database link
- Scalable and flexible, self-created visualization of the S7 program
- Versatile and high-performance recipe and data logging applications.

2.3 Alternative solutions

Alternative software solutions

The mechanisms described above offer the integration of any type of data link modules into the control cycle of WinAC RTX, such as:

- Batch components (PCS7)
- Archiving tools (e.g.: WinCC)
- Operator control and monitoring with classic means, e.g. WinCC
- Use of the product Siemens Industrial Data Bridge instead of the Bridge Client
Alternative hardware solutions

Hardware solutions by Siemens or other manufacturers can also communicate with the OPC server, for example:

- Linux PC (via SOAP)
- Windows PC
- Computers connected via intranet or internet
2.4 Required components

The hardware and software components for the respective stations are listed in the following tables. You can order the components listed in the tables directly in the Siemens A&D Mall at www.ad.siemens.com/mall.

Hardware components for industrial PC

For this application you only require an industrial PC.

Table 2-6

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty.</th>
<th>MLFB / Order number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial PC SIMATIC Rack PC IL 40 S</td>
<td>1</td>
<td>6AG4011-0AA21-0KX0</td>
<td>Configurator: See FAQ ID 17128155</td>
</tr>
<tr>
<td>CP 5613 A2 communications processor for PROFIBUS, PCI card</td>
<td>1</td>
<td>6GK1 561-3AA01</td>
<td>Required only if client programming is to be performed via PROFIBUS</td>
</tr>
<tr>
<td>Ethernet cable, cross-over</td>
<td>1</td>
<td>Depending on manufacturer</td>
<td>Only if client is to run on the PG/PC or if programming is to be performed via Ethernet.</td>
</tr>
</tbody>
</table>

Hardware components for the PG/PC

Table 2-7

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty.</th>
<th>MLFB / Order number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming device Power PG</td>
<td>1</td>
<td>6ES7751-.....-....</td>
<td>Configurator: See FAQ ID 17128155; CP 5611 integrated</td>
</tr>
</tbody>
</table>

Software components of the SIMATIC PC station (WinAC station)

Either an MS Access database or a MySQL database can be used.

Table 2-8

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty.</th>
<th>MLFB / Order number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMATIC WinAC RTX V4.1</td>
<td>1</td>
<td>6ES7671-0RC04-0YA0</td>
<td>SIMATIC NET CD is included in the WinAC package.</td>
</tr>
<tr>
<td>Internet Information Services (IIS)</td>
<td>1</td>
<td></td>
<td>Included in the Windows operating system (only Windows 2000 Prof. / Windows XP Prof.)</td>
</tr>
</tbody>
</table>
### Part A1: Application Description

**Automation Solution**

**Database Link to WinAC via OPC-XML**

**Entry ID:** 21576581

<table>
<thead>
<tr>
<th>Software</th>
<th>Required?</th>
<th>Description</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET Framework V1.1</td>
<td>1</td>
<td></td>
<td>Can be downloaded at <a href="http://go.microsoft.com/fwlink/?LinkId=9832">http://go.microsoft.com/fwlink/?LinkId=9832</a></td>
</tr>
<tr>
<td>MySQL V4.0.20D for Windows (optional)</td>
<td>1</td>
<td>Free of charge (please note the license agreement!)</td>
<td>Can be downloaded at <a href="http://www.mysql.de/">http://www.mysql.de/</a> (please download the server version)</td>
</tr>
<tr>
<td>MySQL Control Center (optional)</td>
<td>1</td>
<td>We recommend installing this program for effective creation and processing of MySQL databases.</td>
<td>Can be downloaded at <a href="http://www.mysql.de/">http://www.mysql.de/</a></td>
</tr>
</tbody>
</table>

---

© Siemens AG 2005. All rights reserved.
Software components of the PG/PC

Table 2-9

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty.</th>
<th>MLFB / Order number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP 7 V5.3 .NET Framework V1.1</td>
<td>1</td>
<td>6ES7810-4CC07-0YA5</td>
<td>Free of charge</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td>Can be downloaded at <a href="http://go.microsoft.com/fwlink/?LinkId=9832">http://go.microsoft.com/fwlink/?LinkId=9832</a> (optional)</td>
</tr>
<tr>
<td>Microsoft Visual Studio .NET 2003 Professional (optional)</td>
<td>1</td>
<td></td>
<td>Can be ordered via your administrator or at <a href="http://www.microsoft.com/">http://www.microsoft.com/</a></td>
</tr>
<tr>
<td>ByteFx .NET Data Provider Library (optional when using MySQL and for recompilation of the C# code)</td>
<td>1</td>
<td>Free of charge (please note the license agreement!)</td>
<td>Can be downloaded at <a href="http://www.bytefx/">http://www.bytefx/</a></td>
</tr>
</tbody>
</table>

Example project

The example application described in this document consists of the following components:

Table 2-10

<table>
<thead>
<tr>
<th>Component</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>21576581_WinAC_SO_CODE_v10_e.zip This file contains:</td>
<td>This file includes all ZIP files listed on the left. It comprises all application codes and the associated data (STEP 7 program for WinAC RTX, source code for Visual Studio C#.NET and databases)</td>
</tr>
<tr>
<td>• STEP7_Ppj.zip</td>
<td>S7 project with configuration and code, which can be directly loaded to WinAC RTX.</td>
</tr>
<tr>
<td>• Client_Application.zip</td>
<td>Executable file of the client application</td>
</tr>
<tr>
<td>• Client_Source.zip</td>
<td>Source code of the client application.</td>
</tr>
<tr>
<td>• RecipeAdministration.mdb</td>
<td>ACCESS database file</td>
</tr>
<tr>
<td>• MySQL_DB.zip</td>
<td>MySQL data folder</td>
</tr>
<tr>
<td>21576581_WinAC_SO_DOKU_v10_e.pdf This document</td>
<td></td>
</tr>
</tbody>
</table>
Effort required for programming/configuring

If you intend to create a complete comparable application, the periods of time listed below have to be planned for programming and configuration. If you use this application as a basis, the application is ready for use after a maximum of two days.

- Assuming average SIMATIC knowledge, 12 man-days are required for the programming of the S7 program to implement a comparable application.
- Assuming good Windows programming knowledge, programming the client application takes approx. 19 man-days.
- Assuming knowledge of databases, the creation of databases takes approx. 2 days.
- Approx. 6 days are to be estimated for the integration of all components.

Note

The periods of time estimated above are rough guide values and may be considerably longer or shorter depending on the previous knowledge.
## 3 Performance Data

### Performance data of system software and configuration

The table below informs on the performance data of the system software and configuration. You are provided with an overview of the performance of this application and its components.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission time of recipe data from Access database $\rightarrow$ WinAC</td>
<td>Approx. 4s max. 20 individual steps (14 bytes per step) in this application (can easily be extended).</td>
<td>Depends on the processor speed, running software, recipe size and type of connection to the database (local/distributed) or to WinAC (local/distributed).</td>
</tr>
<tr>
<td>Transmission time of recipe data from MySQL database $\rightarrow$ WinAC</td>
<td>Approx. 3s max. 20 individual steps (14 bytes per step) in this application (can easily be extended).</td>
<td>Depends on the processor speed, running software, recipe size and type of connection to the database (local/distributed) or to WinAC (local/distributed).</td>
</tr>
<tr>
<td>Transmission time of quality data from WinAC database</td>
<td>&lt; 1s (depending on the amount of data)</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

The maximum size of a data block in WinAC RTX is 65536 bytes; this results in a maximum number of 4680 individual steps (1 step $\times$ 14 bytes). If more steps are required, STEP 7 and C# program have to be changed extensively.
Part A2: Function Mechanisms

Overview

Contents of Part A2

Part A2 provides a detailed description of the function processes of the hardware and software components involved. It is only required to read this part if you are interested in the detailed process and the interaction of the solution components.

It provides information on the solution structure of the application. This part of the documentation includes e.g. basics and functions of the ADO.NET interface.

Objectives of Part A2

Part A2 of this document provides the reader with the following:

- Explanation of all integrated function elements
- Description of the components which can be easily integrated into your specific applications

Topics

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Function Mechanisms</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Basics: PC-based control with SIMATIC</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Web services, IIS and OPC XML-DA</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>Microsoft .NET platform</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>Basics of the C# programming language</td>
<td>30</td>
</tr>
<tr>
<td>4.5</td>
<td>Basics of the ADO.NET interface</td>
<td>31</td>
</tr>
<tr>
<td>4.6</td>
<td>Basics of ACCESS</td>
<td>35</td>
</tr>
<tr>
<td>4.7</td>
<td>Basics of MySQL</td>
<td>36</td>
</tr>
</tbody>
</table>
4 Function Mechanisms

What is the purpose of this chapter?

Basically, this application can be used immediately. The installation instructions tell you how to start the application without reading this chapter. However, if you want to understand what happens behind the scenes, we recommend reading this chapter. You also require in-depth information if you want to adapt specific modules of the application to your requirements.

4.1 Basics: PC-based control with SIMATIC

For the basics of the PC-based SIMATIC WinAC RTX controller, please refer to document /1/.

4.2 Web services, IIS and OPC XML-DA

The basics of web services, the Internet Information Services (IIS) and the OPC XML-DA interface are available in document /2/.

4.3 Microsoft .NET platform

Overview

.NET is a Microsoft technology which enables software development independent of the specific platform to the greatest possible extent. Common Language Runtime (CLR) is used as runtime environment for the different .NET languages (C#, C++, etc.). During compiling, the program code is not converted into a machine-readable binary code but into an Intermediate Language (IL).

When a program is started, this IL code is compiled into a machine-related code by the JIT (Just in Time) compiler.

Microsoft Visual Studio .NET 2003 is used as programming environment for .NET. The Bridge Client of this application has been developed completely with this environment.

Advantages

A Windows application programmed in .NET is not restricted to use on a PC, it can also be used on other devices (e.g.: Panels with Windows CE and .NET Framework). In addition, simple and clear database links and web applications (ASP.NET) can be created for .NET applications.
The figure below shows an overview of the components used in .NET.

Figure 4-1

.NET platform

4.4 Basics of the C# programming language

Introduction

The C# programming language was specially developed for the use of .NET Framework and combines the advantages of different programming languages. Some syntax elements are very similar to C++ and Java, others were been taken from Visual Basic (e.g.: SWITCH). C# is an object-oriented programming language.

Differences between C# and C++

Previously, most Windows applications were developed with the C++ programming language. The following section provides a rough comparison of the two programming languages C# and C++ to illustrate the performance capability and the easy handling of the C# programming language.

The differences stated below refer to C++ in “unmanaged mode”. This mode was the native programming for Windows applications before the introduction of .NET. If C++ is executed in “managed mode” based on the .NET library, its properties are similar to those of C#.
### 4.5 Basics of the ADO.NET interface

The following section focuses on the ADO.NET interface. The reader requires basic knowledge in the field of databases.

**Introduction**

Basically, databases consist of one or more interrelated tables which can be linked correspondingly.

The ADO.NET interface was developed to enable access to these databases in self-created programs. This interface offers uniform procedures, irrespective of the type of database.

ADO.NET, an abbreviation of “Active Data Object”, is the database access interface based on .NET.

In contrast to ADO, the data are no longer stored in a RecordSet but in a DataSet which is stored internally using an XML structure. This method offers the advantage of significantly easier and clearer data processing and organization.

ADO.NET uses two different methods for data exchange:

- Connected data access and
- disconnected data access.
Schematic illustration of the two data exchange methods

These two methods will be explained in the following.

**Connected data exchange**

Connected data exchange includes all connections to the database. The connections are realized by means of a data adapter which is provided by the respective .NET data provider. Data providers are libraries which make available standard ADO.NET functions for the different databases.

They enable a uniform procedure for communicating with the database. The data adapter provides different types of commands. It establishes the connection to the database only if required and closes the connection directly after transferring the data. After successful connection buildup, the data can be transferred to a DataSet. A DataSet is a special storage location for one or more database tables in the main memory of the PC.

**Unconnected data exchange**

Unconnected data exchange occurs after transferring the data from the database to the Dataset. The data in the Dataset are accessed without online connection to the database. After processing, the Dataset can either be transferred back to the database or stored in an XML file.
Provider

A .NET data provider provides all ADO.NET functionalities for a specific database. Standardized interfaces enable identical programming of applications irrespective of the used database.

DataSets

A DataSet contains a specific amount of data from a database (consisting of one or more data tables). Access to these data in the DataSet follows a specific structure as defined in the DataSet’s properties.

DataSets can be defined as typed or as non-typed data sets. A typed DataSet provides the tables and columns of a database directly as properties with the names used in the database. The advantage of this variant is that the table names can already be used when creating the database.

Example: In the “DataSetFull” DataSet, an element of the “Expiry” table in row “1” and column “Method” could be accessed with the following statement:

```csharp
DataSetFull.Expiry[1].Method
```

Non-typed DataSets, however, are addressed with indices. A call for the same element would read as follows:

```csharp
DataSetFull.Tables[“Expiry”].Rows[1].ItemArray[1];
```

This type of addressing offers more flexibility, but has the disadvantage that the location of the corresponding columns in the ItemArray must be defined already when creating the database.

Advantages of using DataSets

DataSets enable off-line data processing of parts of a database. The main advantage of using a DataSet is the improved scalability of the database application. Connection to the database server involves a series of administrative information to be managed by the database management system and, depending on the type of used hardware and software components, a maximum permissible number of connections exists.

If a concept for online data processing is used, the connection between clients and server is maintained while the tasks are being processed. Since the management of the connections uses server resources, the number of maximum connections is limited. Usually, the maximum number of connections corresponds to the number of clients operating with data from the database.

With a .NET concept (off-line data processing), however, a considerably larger number of clients can be linked to the database.
Example:

Assuming an average client-server connection time of 5% of the overall processing time, the number of clients which can access the database is twenty times higher than when using the online connection method.

DataAdapter

A DataAdapter is used to establish the connection between database and DataSet, to terminate it and to perform additional operations such as update, insert, etc. The database is accessed by means of SQL commands. If a DataSet is to be written back to the database, it is possible to use the fill method of the adapter. However, this only works if one table is used per DataAdapter. However, several DataAdapters can be allocated to one DataSet.

Connections

Connections are configured for one or more DataAdapters for connection to the respective database. The DataAdapter opens and closes the connections as required.

Problem – optimistic locking

This new approach with unconnected data processing also involves disadvantages. This becomes clear if you want to write processed data back to the database with the DataSet. If the same table is simultaneously processed by another client and if the modified table is written to the database before the data record of our DataSet has been retransmitted, an error message will be displayed during the next update with the DataAdapter.

The next step would be an analysis of the error and the DataSet would have to be checked for compliance with the current data records of the database. This type of data processing is called “optimistic locking”.

In contrast to offline data processing, online processing as previously used in the database world first checks for possible errors and subsequently performs a transaction. This procedure is referred to as “pessimistic locking”.

In this application, this problem was avoided by reducing the write procedures to log-relevant functions which are used for continuous data logging. Using an auto-increment functionality in the ID fields of the database and by omitting these fields in the DataSet, an Insert SQL command is transferred to the database during updating, so that the ID is always automatically incremented.

Note

In case of more extensive data manipulations based on the DataSet, it is absolutely necessary to implement an exact handling of exceptions.
4.6 Basics of ACCESS

Differences between desktop database and database with management system

This application example offers two alternative databases: A desktop database (ACCESS) and a database with database management system (MySQL).

At first glance, the two database types show only minor differences. On closer examination, however, important differences become clear. The management program of a desktop database is executed on each distributed station. The fact that one common database is accessed may cause conflicts.

When using a database with management system, all workstations indirectly access the database via the management program (server). The server solely manages all accesses to the database. Consequently, access conflicts cannot occur.

Why ACCESS?

ACCESS is a universal desktop database for office work on a small scale. This results in the problem that it provides less performance capability than a high-performance SQL database with a server-client architecture.

However, ACCESS is integrated into the Microsoft Office Suite and thus already installed on many office workstations and immediately available without extra installation work. Furthermore, ACCESS offers the advantage of creating databases graphically and easily understandable and of filling these databases with data. It also includes special export functions enabling to convert databases created in ACCESS also into other databases. (E.g. into an MS SQL database)

For users with basic knowledge of databases, ACCESS is self-explanatory and easy to use.

For detailed information on the use of ACCESS, please refer to the introduction in the online help.

ACCESS under ADO.NET

ACCESS databases can be accessed under ADO.NET via an oleDB data provider (with Microsoft Jet Engine). The data can then be processed with the aid of DataSets.

Visual Studio includes a wizard for linking databases for easy integration of ACCESS databases into Windows applications.

For detailed information on the use of the wizard, please refer to 7.3.3 Database accesses (C# Windows program).
4.7 Basics of MySQL

Introduction
MySQL is an SQL database system which is available on a large number of platforms. MySQL offers functionalities which are otherwise only provided by very expensive database systems. Since this system is offered under open source conditions, MySQL is becoming increasingly popular.

Function principle of MySQL
MySQL can be installed on a large number of platforms. Our application example, however, merely refers to a Windows-based version. The computer used for database management must be provided with an MySQL server.

See Chapter 5.4 Installation of the MySQL database server.

On the computer with the OPC client, at least the client software (Bridge Client described above) must be installed. It is also possible to install the OPC server and the MySQL server on the same computer. See Chapter 5.7 Installation of the Windows Bridge Client.

Notes on the license agreement
Up to now, MySQL AB has published the MySQL server for two license versions – a GPL (General Public License) and a commercial license (low price). With the GPL version, the source code is freely available and can be used for own projects, provided that these projects are later again published with source code. The client (DLL library) has previously been published with an LGPL (Less General Public License). LGPL is a less restrictive license which allows publication in own projects without disclosing the source code.

Currently, MySQL AB is thinking about publishing the client also under GPL conditions. For our customers this would mean that they must either disclose the program project or purchase a server license.

Note
If you want to use a MySQL database for the storage of your control data, make sure to comply with the exact license conditions of the manufacturer.
Part B: Installation of the Example Application

Overview

Contents of Part B

Part B takes you step by step through installation and start-up of the application.

Objectives of Part B

Part B of this document provides the reader with following:

- Explanation of the example with all hardware and software components
- Illustration of the operation of the application

Topics

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Installation of Hardware and Software</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>Preparatory installation</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Hardware installation</td>
<td>39</td>
</tr>
<tr>
<td>5.3</td>
<td>Installation of WinAC RTX</td>
<td>41</td>
</tr>
<tr>
<td>5.4</td>
<td>Installation of the MySQL database server</td>
<td>42</td>
</tr>
<tr>
<td>5.5</td>
<td>Installation and setup of the Internet Information Services</td>
<td>42</td>
</tr>
<tr>
<td>5.6</td>
<td>Installation of the STEP 7 project</td>
<td>46</td>
</tr>
<tr>
<td>5.7</td>
<td>Installation of the Windows Bridge Client</td>
<td>47</td>
</tr>
<tr>
<td>5.8</td>
<td>Configuration and programming of WinAC</td>
<td>47</td>
</tr>
<tr>
<td>5.9</td>
<td>Configuration of the MySQL database</td>
<td>48</td>
</tr>
<tr>
<td>5.10</td>
<td>Configuration and programming of the ACCESS database</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>Operating the Application</td>
<td>49</td>
</tr>
<tr>
<td>6.1</td>
<td>Operating the MySQL database</td>
<td>49</td>
</tr>
<tr>
<td>6.2</td>
<td>Operating the Windows application (Bridge Client)</td>
<td>52</td>
</tr>
<tr>
<td>6.3</td>
<td>Operating the S7 control program</td>
<td>56</td>
</tr>
</tbody>
</table>
5 Installation of Hardware and Software

5.1 Preparatory installation

Introduction

It is absolutely necessary to install STEP 7 and the SIMATIC NET software before installing WinAC RTX. We strongly recommend adhering to the installation sequence described below.

Notes

Please install the listed software components on a newly installed Windows system.

Before using this application in a real process, please consider the aspects of network security. For better comprehensibility, security aspects are not part of this application description.

STEP 7

STEP 7 is installed on the PG/PC which is intended for the configuration and the programming of the automation stations. Alternatively, you can install STEP 7 on the PC (WinAC station) on which WinAC is to run.

At this point, the installation of STEP7 will not be described. The installation is self-explanatory and performed in the usual Windows environment.

SIMATIC NET

The SIMATIC NET PC software is installed on the same PC (WinAC station) on which WinAC is to be installed. The software package includes all tools required to install and operate a PC station.

From STEP 7 version 5.2 on, “Advanced PC Configuration” is used to put PC stations into operation. This enables the configuration of PC stations directly in STEP 7. Before using “Advanced PC Configuration”, we strongly recommend reading /4/.

Microsoft Visual Studio .NET 2003

If you want to adapt the Windows application (Bridge Client) to your specific requirements, please install Microsoft Visual Studio .NET 2003 on your PG/PC or on the industrial PC before changing the application. Insert installation CD 1, start the Setup.exe file and follow the instructions.

Installation of Byte FX .NET DataProvider

If you want to modify the C# Windows application (Bridge Client), Byte FX.NET DataProvider has to be installed on your computer. If you only want to execute the Bridge Client, the following installation is not required.
Table 5-1

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If not already done, please download the Byte FX library which can be downloaded at <a href="http://www.bytefx.com">http://www.bytefx.com</a>.</td>
</tr>
<tr>
<td>2</td>
<td><strong>Extract</strong> the zip archive and execute the &quot;<strong>Setup.exe</strong>&quot; file. Follow the instructions.</td>
</tr>
</tbody>
</table>

**Note**

Before using this provider, please make sure you have studied the license stipulations applicable for this library. Alternatively to this provider, an ODBC driver for MySQL can also be used.

**Attention**

This library has to be installed also if you do not intend to use the MySQL database link; otherwise errors will occur during compiling the Windows application. (Applies only if you want to recompile the source code)

5.2 Hardware installation

### Installation of the IPC

Install the PC as described in the installation instructions included in the delivery of the PC. If the client and the databases are installed on the PG, connect the PC to the PG/PC using a cross-over TCP/IP cable. Subsequently, follow the instructions described in the following section.

If you use the industrial PC for data storage and operation of the Bridge Client, you can skip the following section.

### Setting the Ethernet address

If you use a remote client, set the following addresses on both Ethernet cards.
### Part B: Installation of the Example Application

#### Installation of Hardware and Software

**Database Link to WinAC via OPC-XML**

**Entry ID: 21576581**

**Table 5-2**

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the “Start” menu of the respective computer, click the “My Network Places” button.</td>
</tr>
<tr>
<td>2</td>
<td>Now click the “View network connections” button.</td>
</tr>
<tr>
<td>3</td>
<td>Select the connection you have previously wired with the right mouse button and then click “Properties”.</td>
</tr>
<tr>
<td>4</td>
<td>In the “Properties” dialog box, double-click “Internet Protocol (TCP/IP)”.</td>
</tr>
<tr>
<td>5</td>
<td>In the “IP address” field, enter the address 192.168.115.10 for the industrial PC and the address 192.168.115.20 for the PG/PC. If not yet automatically assigned, enter “255.255.255.0” as subnet mask. <strong>Confirm</strong> both dialog boxes by clicking “OK”, confirm the safety queries and close the windows.</td>
</tr>
</tbody>
</table>
Note

The procedure described above should only be used for intranets or small networks, since all safety precautions have been neglected. If you work in a company network (intranet, etc.), please consult your system administrator before changing your network settings!

Installation of CP 5613

Install CP 5613 in an available PCI slot of the rack PC as described in the installation instructions /8/ included in the delivery. The CP drivers have already been included in the installation of the SIMATIC NET PC software.

For further information on the installation of PCI cards in rack PCs, please refer to PC manual /7/.

5.3 Installation of WinAC RTX

Note

Administrator rights ("ADMIN") for your operating system are required for the installation of WinAC RTX V4.0.

The installation of WinAC RTX is described in the following table.

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The <strong>setup program starts automatically</strong> after inserting the WinAC RTX CD. If the program does not start, execute the &quot;Setup.exe&quot; program on the CD.</td>
</tr>
<tr>
<td>2</td>
<td>After selecting the language, a dialog box is displayed which guides you through the installation tasks.</td>
</tr>
<tr>
<td>3</td>
<td>Click the <strong>&quot;install VenturCom RTX&quot; button</strong> and follow the instructions in the dialog box. Ardence RTX (formerly VenturCom RTX) is installed on your PC. <strong>Note</strong> The license number (Runtime PAC Number) and the e-mail address for the licensing of Ardence RTX are on the rear of the WinAC RTX CD cover. <strong>Note</strong> If the error message &quot;Your System is using a HAL that is not supported by RTX 5.12&quot; is displayed during the installation of the Ardence RTX extensions please read the FAQ with the ID 17053416 on the A&amp;D Support home page (<a href="http://www.ad.siemens.com/support">www.ad.siemens.com/support</a>).</td>
</tr>
<tr>
<td>4</td>
<td>After restarting the PC, the functionality of the Ardence RTX extensions has to be checked. <strong>Click</strong> the <strong>&quot;Step 2&quot; circle</strong> in the <strong>&quot;WinAC RTX V4.0 Setup&quot;</strong> dialog box. You receive instructions for testing the RTX extensions.</td>
</tr>
<tr>
<td>5</td>
<td><strong>Click</strong> the <strong>&quot;Step 3&quot; circle</strong> in the installation dialog box to start the installation of the WinAC RTX software. Then follow the instructions of the setup program.</td>
</tr>
</tbody>
</table>
5.4 Installation of the MySQL database server

Installation and configuration of MySQL

For correct installation of MySQL, please proceed as follows:

Table 5-4

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Download the latest MySQL version at <a href="http://WWW.MySQL.DE">WWW.MySQL.DE</a>.</td>
</tr>
</tbody>
</table>
| 2   | Extract the ZIP file with an unpacking program.  
   (in Windows XP, click the folder with the right mouse button and select “Extract all files”, then follow the instructions). |
| 3   | In the extracted folder, double-click the “Setup.exe” file.  
   Follow the instructions, create a directory for installation and write down this directory. |
| 4   | If MySQL has already been installed, you first have to delete the My.ini file from the Windows directory.  
   (This file is newly created when the server is started for the first time). |
| 5   | Change to the selected directory, select the “BIN” folder and start the file “WinMySQLAdmin.exe”. |
| 6   | The dialog box displayed on the right appears. Enter a user name and a password and click OK to confirm your entries. |
| 7   | A small red traffic light is now displayed in the information bar which shows the status of the MySQL server.  
   - Red indicates that the server is in stop mode.  
   - Green indicates that the server has been started. |
| 8   | Click the traffic light symbol with the right mouse button and select “Win NT -> Install the Service”.  
   Confirm the query.  
   The MySQL server is now installed as Windows Service.  
   If this item is not offered in the menu, the service has already been installed.
Part B: Installation of the Example Application

Installation of Hardware and Software

Database Link to WinAC via OPC-XML

Entry ID: 21576581

9. Reopen the context menu of mysqladmin, select “Win NT -> Start the Service” and confirm the query.

The traffic light symbol should now light up green (server started).

10. Again click the traffic light symbol with the right mouse button and select “Show me”.

11. This dialog box opens.

12. Click the my.ini Setup button.

Click the “Create Short Cut on Start Menu” button. The administration tool is now started each time the system is started and the traffic light symbol always displays the current server status.

13. Click the traffic light symbol in this dialog box with the right mouse button and select “hide me” to place the dialog box in the information bar.

The MySQL server is now installed and ready for use. For further information on handling and programming, please refer to Chapter 5.9 Configuration of the MySQL database.
5.5 Installation and setup of the Internet Information Services

The Internet Information Services (IIS) offer the functionality for providing web services. For this reason, the ISS are installed on the PC on which the OPC server is installed – thus on the WinAC station.

Installation

To install the IIS, follow the steps listed below. Have your Windows installation CD ready. It is absolutely necessary that the IIS are installed before installing Microsoft .NET Framework 1.1. If you have already installed .NET Framework, please observe the note at the end of the installation table.

Table 5-5

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open the “Add or Remove Programs” dialog box via: Start ➔ Settings ➔ Control Panel ➔ Add or Remove Programs</td>
</tr>
</tbody>
</table>
| 2   | Select “Add/Remove Windows Components”.
| 3   | In the component list, select “Internet Information Services (IIS)” by checking it. |
| 4   | Change to the “Internet Information Services (IIS)” dialog box by clicking “Details”.
| 5   | Only select the following entries in the “Subcomponents” list:
|     | • Common Files
|     | • Internet Information Services Snap-In
|     | • Personal Web Manager (only Windows 2000)
|     | • WWW server
|     | Confirm your selection by clicking “OK”.
| 6   | Start the installation by clicking “Next >” and follow the instructions of the installation program. |
| 7   | If your WinAC station is connected to the internet, check whether important updates for the IIS are available at http://windowsupdate.microsoft.com/. Available updates can also be downloaded and installed manually at www.microsoft.com/downloads. |

Note

If you have already installed “Microsoft .NET Framework 1.1” before installing IIS (Framework is listed in the “Add or Remove Programs” dialog box in the control panel) perform the following steps:

1. Open the DOS prompt via: Start ➔ Run ➔ Enter “cmd” ➔ OK
2. Enter "%systemroot%\Microsoft.NET\Framework\v1.1.4322\aspnet_regiis –i" and press the return key.
3. Restart the PC.
Setup

To enable the access of applications to the SIMATIC NET OPC server via OPC XML-DA, the Internet Information Service has to be configured accordingly. The following table explains how to set up the IIS for this access.

The default settings are kept for access protection. This means that each user can access the web service. That is the reason why you should operate this example only in an isolated network or intranet.

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open the “Internet Information Services” via: Start → Control Panel → Administrative Tools → Internet Services Manager</td>
<td><strong>Note</strong> If Administrative Tools is not available proceed as follows: Start → Run and enter the following string: %SystemRoot%\System32\Inetsrv\iis.msc</td>
</tr>
<tr>
<td>2</td>
<td>In the context menu of the branch “Default website” (right mouse click) select → New → Virtual folder</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Click the “Virtual Directory Creation Wizard” and then the “Next” button.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>In the “Alias:” box, enter “OPC.Simatic.NET” and click “Next &gt;”.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>By clicking the “Browse” button, select the subdirectory “opc2\binXML” in your SIMATIC.NET installation directory. This directory includes the “sopcweb.asmx” file. Click the “Next &gt;” button.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>In the “Access Permissions” window, only select the following points: • Read • Run scripts (such as ASP) Click the “Next &gt;” button.</td>
<td></td>
</tr>
</tbody>
</table>
### 5.6 Installation of the STEP 7 project

For information on the configuration of WinAC RTX and of the Station Configuration Editor, please refer to document /1/.

#### Installing project

Proceed as follows to open the STEP 7 project and to adapt it to your configuration.

Table 5-7

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
<th>Note / Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Please configure the Station Configuration Editor on the PC station (industrial PC) as described in Chapter 4.7 of document /2/. Select an OPC server under index 1 and WinAC RTX under index 2 for the station.</td>
<td>Optionally, you can additionally install an IE (Industrial Ethernet) adapter on index 3 (adapting the MAC address in HW Config required). The listed installation of the PROFIBUS IF module is also optional.</td>
</tr>
<tr>
<td>2</td>
<td>Open the SIMATIC Manager.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>Retrieve</strong> the archive 21576581_WinAC_SO_CODE_v10_e.zip via the menu &quot;File &gt; Retrieve...&quot;.</td>
<td>You can open the project immediately after retrieving.</td>
</tr>
</tbody>
</table>
### 4. Instruction

**Start WINLC RTX by selecting**

START > SIMATIC > PC BASED Control

**WINLC RTX**

**Open the hardware configuration** 

(double-click Configuration in the PCWinAC folder) and download it to WINLC

Now please select all blocks in the block container except the two variable tables and the UDT1 structure, then click the "Download to PLC" button.

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
</table>
| 4   | Start WINLC RTX by selecting START > SIMATIC > PC BASED Control WINLC RTX  
|     | **Open the hardware configuration** 
|     | (double-click Configuration in the PCWinAC folder) and download it to WINLC  
|     | Now please select all blocks in the block container except the two variable tables and the UDT1 structure, then click the "Download to PLC" button. |

**Note**

Make sure that the station name PCWinAC is set in the Station Manager.

---

### 5.7 Installation of the Windows Bridge Client

The installation is performed using the Setup.exe file located in the Client_Application.zip archive. Start the file and follow the program instructions.

**Note**

Checking the client functionality requires the installation of Microsoft .NET Framework.

### 5.8 Configuration and programming of WinAC

WinAC is configured and programmed as described in document /1/. 
5.9 Configuration of the MySQL database

Installation of the MySQL database

To install the MySQL database included in the delivery, please proceed as follows:

Table 5-8

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Extract the Zip archive “MySQL_DB.ZIP” using an unpacking program. (in Windows XP, click the folder with the right mouse button and select “Extract all files”, then follow the instructions).</td>
</tr>
<tr>
<td>2</td>
<td>Stop the MySQL server as described in Chapter 5.4 Installation of the MySQL database server using the traffic light symbol.</td>
</tr>
</tbody>
</table>
| 3   | Copy the extracted folder to the directory “C:\mysql\data”. (Alternatively to drive c: Select the drive specified by you).  
   | Note: The folder must be the RecipeAdministration folder which was stored in the archive.  
   | If you have selected “Create new folder” before extracting, please select a folder in the lowest directory level.  
   | The folder to be copied must not contain any subfolders! |
| 4   | Click the traffic light symbol to restart the server. |

5.10 Configuration and programming of the ACCESS database

Overview

The ACCESS desktop database included in the delivery of Microsoft Office Professionell features a large variety of functions for creating and processing.

ACCESS provides the following functionalities for the processing of data:

- Creating, editing tables
- Generating queries (views)
- Creating reports
- Web sites, etc.
- Export of the data and structures to MS SQL and other databases

A wizard is available for creating new databases. For detailed information on working with ACCESS, please refer to the online help.

Installation

Copy the ACCESS database file “RecipeAdministration.mdb” to the desired directory.
6 Operating the Application

6.1 Operating the MySQL database

If you want to test the application first, you can skip this chapter. However, we recommend performing a short test of the instructions, since this enables to test the correct functionality of the database.

Working with SQL commands in MySQL

Like any other database system, MySQL features an SQL command window in which the individual SQL statements can be entered or scripts can be executed (a script includes several SQL commands which may exceed the length of the command window).

To send SQL commands to the MySQL server installed on your computer, proceed as follows:
### Part B: Installation of the Example Application

#### Operating the Application

**Database Link to WinAC via OPC-XML**

**Entry ID: 21576581**

**Table 6-1**

<table>
<thead>
<tr>
<th>No.</th>
<th>Note</th>
</tr>
</thead>
</table>
| 1   | Open the “C:\mysql\bin” folder in the **Explorer**.  
Note: The location of the folder depends on the specific system (as defined during installation). |
| 2   | Click the “mysql.exe” file with the **right mouse button** and select **Create Shortcut**. |
| 3   | Click this shortcut with the **right mouse button** and select **Properties**.  
Note: The shortcut is always marked by a small arrow. |
| 4   | **Target**: C:\mysql\bin\mysql.exe -u root  
Supplement the “-u root” string in the **Target** text line as shown above. Now click **OK**. |
| 5   | **Start** the “mysql.exe” file in the “C:\MySQL\BIN” directory .  
Now you are logged in to the database with the user name “root”. |
| 6   | The following MySQL message appears. |
| 7   | Enter the command  

```
Use RecipeAdministration + ENTER
```

to change the database.
If you want to show all columns of the Expiry list, enter the following SQL command:

```
Select * from Expiry;
```

The Expiry table is displayed.

---

**Working with the MySQL Control Center**

In addition to the MySQL server, you should install the MySQL Control Center to conveniently create and process MySQL databases.

The operation of the MySQL Control Center is intuitive and self-explanatory. For effective operation of the MySQL CC, however, basic knowledge in the field of databases is required.

For detailed information on the operation of the Control Center, please refer to the online help or to the MySQL home page

(http://www.mysql.de).
Part B: Installation of the Example Application
Operating the Application

6.2 Operating the Windows application (Bridge Client)

The Bridge Client consists of two Windows sheets which cover the bridging and visualization functions.

Overview – main window

The main window with the settings for the bridge function is structured as follows:

Figure 6-1

The individual controls are combined in a total of 9 element groups.

Controls of the main window

Table 6-2

<table>
<thead>
<tr>
<th>Element group</th>
<th>Element</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC Settings</td>
<td>local</td>
<td>Using the URL for a local OPC-XML access.</td>
</tr>
<tr>
<td></td>
<td>distributed</td>
<td>Using the URL for a remote OPC-XML server.</td>
</tr>
<tr>
<td></td>
<td>OPC Server URL</td>
<td>OPC URL to specify the position of the OPC-XML server.</td>
</tr>
<tr>
<td></td>
<td><a href="http://192.168.115.10/OPC_Simatic.NET/opcweb.xml">http://192.168.115.10/OPC_Simatic.NET/opcweb.xml</a></td>
<td></td>
</tr>
</tbody>
</table>
### Part B: Installation of the Example Application
### Operating the Application

**Database Link to WinAC via OPC-XML**

#### Entry ID: 21576581

<table>
<thead>
<tr>
<th>Recipe Number</th>
<th>Recipe Number enticing the number of the desired recipe is entered.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Mode</td>
<td><strong>Auto</strong> Sets the client to automatic mode. This means that the Bridge Client is fully activated when the Start button is clicked (connection to database, connection to OPC server, etc.). <strong>Manual</strong> Sets the Client to manual mode. This means that the Bridge Client is activated individually and manually. The corresponding buttons are available in the “Manual Mode” group.</td>
</tr>
<tr>
<td>Kind of Database</td>
<td><strong>ACCESS</strong> Defines that the active recipe database is an ACCESS database. <strong>MySQL</strong> Defines that the active recipe database is a MySQL database.</td>
</tr>
<tr>
<td>Auto Mode</td>
<td><strong>Start</strong> Starts the automatic mode of the client. <strong>STOP</strong> Stops the automatic mode of the client.</td>
</tr>
<tr>
<td>Manual Mode</td>
<td><strong>Connect to OPC</strong> This button is used to establish a connection between client and OPC server. <strong>Connect to DB</strong> This button is used to connect the client to the selected database and to fill the DataSet with data. <strong>Start Polling</strong> This button starts the automatic polling of the status values from the controller with the aid of subscriptions. <strong>STOP Polling</strong> This button is used to stop the automatic polling of the status values. <strong>transfer Recipe</strong> Transfers the selected recipe to the controller. <strong>update DB</strong> This button is used to write the DataSet back to the database and to update the individual tables.</td>
</tr>
</tbody>
</table>
### ACCESS Connection

<table>
<thead>
<tr>
<th>Position</th>
<th>This text box indicates the position of the ACCESS database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>D:\Nicks daten\Qu</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Search</th>
<th>Clicking this button opens a &quot;Open File&quot; dialog box in which the ACCESS database can be found more easily.</th>
</tr>
</thead>
<tbody>
<tr>
<td>search</td>
<td>search ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Name</th>
<th>This text box is used to enter the user name for access to an ACCESS database. (Remains empty in this application example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>user name</td>
<td>user name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Password</th>
<th>Box in which the password is entered. (Remains empty in this application example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>password</td>
</tr>
</tbody>
</table>

### MySQL Connection

<table>
<thead>
<tr>
<th>Server Name</th>
<th>The MySQL server name for access to a MySQL database is entered in this box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>server name</td>
<td>192.168.115.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name of the Database</th>
<th>Enter the name of the database in this text box. The name is RecipeAdministration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name of the database</td>
<td>name of the database</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Name</th>
<th>In this text box, the user name for access to a MySQL database can be entered. (root)</th>
</tr>
</thead>
<tbody>
<tr>
<td>user name</td>
<td>user name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Password</th>
<th>This text box is used to enter the password for access to the MySQL database. (Remains empty in this application example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>password</td>
</tr>
</tbody>
</table>

### PLC Controlling

<table>
<thead>
<tr>
<th>Start S7 Program</th>
<th>Starts the processing of the recipe in S7.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start S7 Program</td>
<td>Start S7 Program</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STOP S7 Program</th>
<th>Stops the processing of the recipe in S7.</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOP S7 Program</td>
<td>STOP S7 Program</td>
</tr>
</tbody>
</table>

### VIEW

<table>
<thead>
<tr>
<th>VIEW</th>
<th>Clicking this button opens the visualization of the process.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIEW</td>
<td>VIEW</td>
</tr>
</tbody>
</table>

---

V 1.1 Edition 05/17/05 54/101
Hauptfenster

The status bar displays the following client information:
- Status of the S7 program
- Currently processed step
- Status of the OPC connection
- Status of the call-back function
- Database connection status
- Current date / time

Overview – visualization

Clicking the VIEW button in the main window opens the visualization window. The screen shot below shows an overview of this window.

Figure 6-2

Controls of the visualization window

Table 6-3

<table>
<thead>
<tr>
<th>Element group</th>
<th>Element</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization</td>
<td>current step nr.</td>
<td>Displays the currently processed step.</td>
</tr>
<tr>
<td></td>
<td>current recipe</td>
<td>Displays the current recipe.</td>
</tr>
<tr>
<td></td>
<td>passed time</td>
<td>Displays the time already required for the recipe.</td>
</tr>
</tbody>
</table>
Using this animated graphic, you can determine the status of the tank. (Water level, temperature, mixer in operation, etc.)

Recipe DB

This table shows all recipes available in the database. While the S7 program is active, it additionally highlights the currently processed step.

Operation

This button is used to start the S7 program in the controller.

This button is used to stop the program in the controller.

The timer interval is used to define how often new visualization data are to be polled at the OPC server.

Main window

This button starts the automatic polling of the visualization data.

Click this button to stop the automatic polling of the OPC server.

These two bars indicate the timer activity (upper bar) and the asynchronous call of the call-back function.

6.2.1 Automatic mode

Procedure

If you want to start the recipe processing in automatic mode, please proceed as follows:

Table 6-4

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
</table>
| 1   | Preparatory steps:  
|     | • If not already done, please perform the installations listed in Chapter 5  
|     |   Installation of Hardware and Software.  
|     | • If WinAC RTX is not yet started, please start it now (see Chapter 5 in document /1/).  
| 2   | Start the executable ClientBridge.exe file under “Start>Siemens Software>ClientBridge”. |
3 Enter a valid URL in “OPC Server URL”.

Note:
- If the client is installed on the same computer as WinAC, the URL is to be set as follows: http://localhost/OPC.Simatic.NET/sopcweb.asmx.
- If the client is installed on a different computer, “localhost” must be replaced by an IP address (e.g. name of the computer), e.g.: http://192.168.115.10/OPC.Simatic.NET/sopcweb.asmx
- Instead of manual entry, you can also use the buttons “local” or “distributed”.

4 Now select the database to be used.
The functionality of the two databases is basically identical.

5 Depending on the used database, the input masks for the connection settings of the respective database are activated.
Enter the corresponding data for the specific database.
Note:
- If you want to use the ACCESS database included in the delivery, you have to search for the RecipeAdministration file in the corresponding directory. Please do not enter a user name and password.
- If you intend to use a MySQL database, please enter the following:
  - server name : localhost (or IP address of the remote computer)
  - name of database : RecipeAdministration
  - user name : root
  - password:
6. **Click** the “Start” button in the “Auto Mode” field.

7. You can now monitor how the individual connections are established.
   - First, the connection to the desired database is established. Subsequently, the data are read. (Status field 5 displays the message “Connected to ACCESS” or “...to MySQL”.)
   - Now, the connection to the OPC server is automatically established and the subscriptions are registered. Status field 3 displays the messages “registering OPC” and “registering subscriptions.”

After registration of the subscriptions, the client changes to polling mode, which automatically polls the status values. The polling mode is active when a bar in status field 4 is in continuous motion.

8. **Enter** the number of the desired recipe in the text box of “Recipe Number”.

9. **Click** the “Start S7 Program” button in the “PLC Controlling” field. The client now transfers the start command to the controller. Subsequently, the data are exchanged between controller and client.

   While the controller is active, the status fields 1, 2, 3 and 4 inform you on the program status, the current step, the connection status to the OPC server and the automatic read function.

10. **After completion of the recipe,** all quality and protocol data stored in the DataSet are stored in the respective database. Status field 1 displays the “recipe finished” message.

The client waits for the next recipe request.

To stop the client or to change the parameters, click “STOP” in the “Auto Mode” field.
6.2.2 Manual mode

If you want to adapt the program to your special requirements or if you want to correct errors of the automatic mode, the manual mode of the client is an option to individually activate the functionalities of the client. Please consider that the functions have to be activated in the order described below to avoid dependency errors.

With the procedure displayed in the following table, you achieve the same status of the application as in the automatic mode described above.

Procedure

Table 6-5

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
</table>
| 1   | Preparatory steps:  
|     | • If not yet done, please perform the installations listed in Chapter 5 Installation of Hardware and Software.  
|     | • If WinAC RTX is not yet started, please start it now (see Chapter 5 in document /1/). |
| 2   | Start the executable ClientBridge.exe file under “Start>Siemens Software>ClientBridge”. |
| 3   | Enter a valid URL in “OPC Server URL”. |

Note:

- If the client is installed on the same computer as WinAC, the URL is to be set as follows: \http://localhost/OPC.Simatic.NET/sopcweb.asmx.  
- If the client is installed on a different computer, “localhost” must be replaced by an IP address (or the name of the computer) e.g.: \http://192.168.115.10/OPC.Simatic.NET/sopcweb.asmx.  
- Instead of manual entry, you can also use the buttons “local” or “distributed”.  

Note: If an error message is displayed during one of the automatically processed steps, each process step can be individually performed in manual mode to correct possible errors (e.g.: Wrong addresses, etc.).
4. Now select the database to be used. The functionality of the two databases is basically the same.

5. Depending on the used database, the input masks for the connection settings of the respective database are activated. Enter the corresponding data for the specific database.

Note:
- If you want to use the ACCESS database included in the delivery, you have to select the RecipeAdministration file in the corresponding directory. Please do not enter a user name and password.
- If you intend to use a MySQL database, please enter the following:
  - server name: localhost (or IP address of the remote computer)
  - name of database: RecipeAdministration
  - user name: root
  - password:

6. Click “Connect To DB” in the “Manual Mode” field.

Status field 5 displays the “connected to My...” message. The database was opened and the data are transferred to the DataSet.

7. Click “Connect to OPC”
### Part B: Installation of the Example Application
#### Operating the Application

**Database Link to WinAC via OPC-XML**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>You can now monitor how the connection to the OPC server is established. Status field 3 displays the messages “registering OPC” and subsequently “registering subscriptions”</td>
</tr>
<tr>
<td>9</td>
<td>With the buttons “Start Polling” and “STOP Polling”, you can activate or deactivate the automatic polling of the values.</td>
</tr>
<tr>
<td>10</td>
<td>After activating the polling mode, status field 4 displays a bar which is in continuous motion as long as the polling mode is active.</td>
</tr>
</tbody>
</table>
| 11   | With the “Transfer Recipe” button you can transfer a recipe from the database to the controller for testing.  
**Note:** Precondition for the use of this test function is that the database connection and the OPC server connection have already been established. |
| 12   | The “Update DB” button is used to test the update function of the database, since updating in automatic mode is performed only after completion of a recipe.  
**Note:** Precondition for the use of this test function is that the database connection and the OPC server connection have already been established. |
| 13   | Enter the number of the desired recipe in the text box of “Recipe Number”. |
| 14   | Click the “Start S7 Program” button in the “PLC Controlling” field. Now the client transfers the start command to the controller. Subsequently, the data are exchanged between controller and client. |
| 15   | After completion of the recipe, all quality and protocol data stored in the DataSet are stored in the respective database. Status field 1 displays the “recipe finished” message. |
| 16   | If you want to newly parameterize or stop the client, please click “STOP Polling” in the “Manual Mode” field.  
If you subsequently change the connection parameters, the procedure starting from no. 6 of these instructions has to be repeated step by step. |
6.2.3 VIEW (visualization)

For easy operation and improved overview, a visualization program with easy-to-use basic functionality is integrated in the client. The visualization function is activated as follows:

**Procedure**

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Precondition</strong> for using the visualization function is an active <strong>connection</strong> to the <strong>OPC server</strong> and successful reading of the recipe data. Both conditions can be met via automatic or manual mode of the client.</td>
</tr>
<tr>
<td>2</td>
<td><strong>Start</strong> the control program by clicking “<strong>Start S7 Program</strong>” button.</td>
</tr>
<tr>
<td>3</td>
<td><strong>Click</strong> “<strong>View</strong>” to open the visualization window.</td>
</tr>
</tbody>
</table>
| 4   | ![Visualization Schematic](image)  

The window shown above appears. The left side of the window shows a schematic illustration of the tank with the corresponding associated information. The right side of the window shows the table with the recipes. |
| 5   | In order to avoid an unnecessary increase in the bus load, visualization has not been started immediately; it has to be started by clicking the “**Start Visualization**” button and stopped by clicking “**STOP Visualization**” if it is no longer required. |

Now click the “**Start Visualization**” button.
6.3 Operating the S7 control program

The control program in the WinAC controller can either be controlled using the variable table in the STEP 7 project or alternatively using the control functions of the client.

Operation with the variable table

Operation with the variable table requires that the client is active, that all connections are established and that the data have been loaded from the respective database.
Preparatory steps:

1. If not yet done, please perform the installations listed in Chapter 5, Installation of Hardware and Software.
2. If WinAC RTX is not yet started, please start it now (see Chapter 5 in document /1/).

Open the SIMATIC Manager.

Click the menu items “File → Open”, select the STEP 7 project “WinAC_OPC_XML_ADO_NET28” and confirm your selection.

Double-click the variable table VAT_engl (or VAT_de) in the block folder of the STEP 7 project.
The variable table shown above appears. The variable table is divided into the three sections “Controlling”, “Watching of state information” and “Observing current values of the process”.

First, click the “Watch” button.
In the “Controlling” section, set the variable “DB_Command_Glob”.CurrentRecipe to “1” and click the “Control” button.

Notes
- In “Current Recipe”, the recipe to be processed is selected; in this case recipe 1 is processed.
- Please check whether the recipe is actually available before starting the process!
- The desired recipe is always selected from the controller with higher priority. If no value (0) is set in the controller, the value shown in the “Recipe Number” dialog box of the client is used automatically.

After the desired recipe number has appeared in the status value field of the controller, you can enter “1” in the “Controlling” section of the variable “DB_Command_Glob”.Start and again click “Control”.

Note
For easy control of the values, use the following shortcuts:
“<Ctrl> + 1” for the value “true” and “<Ctrl> + 0” for the value “false”.

You can now monitor how the data are exchanged between controller and client.
- First, the variable “DB_Command_Glob”.DataRequest is set to “1”; the controller signals data request to the client.
- In the next step, the client clears this variable and sets the variable “DB_Command_Glob”.DataRequestAk to “1” as acknowledgement.
- After transfer of the recipe data to the controller, the client sets the variable “DB_Command_Glob”.DataSend.
- The controller acknowledges the reception by setting the variable “DB_Command_Glob”.DataSendAk and starts the recipe processing which is signaled by the variable “DB_Command_Glob”.PLCProgramRun.

During recipe processing, the current process values can be monitored in the third section of the variable table. The controller does not receive data from the client during recipe processing.

After completion of the recipe, the value of the variable “DB_Command_Glob”.CurrentRecipe is reset to “0” and the controller waits for the next start initiated via the Start variable.

Operation with the Windows application
The operation of the S7 program using the client is performed via the main window of the client as well as via the visualization window. For detailed information, please refer to Chapter 6.2.1 Automatic mode.
Part C: Program Description

Overview

Contents of Part C

Part C is of interest if you want to expand or adapt the software to your system.

Objectives of Part C

This part of the documentation provides the reader with the following:

- Explanation of the code details of some core parts of the program
- Notes on useful expansions.

Requirement

This is not an introduction to the STEP7 languages STL, FBD or the object-oriented language C#. Readers should be familiar with the basics of these programming languages.

Before reading the code description, it is useful to read the chapters in Part A1 and A2.

Topics

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Explanation of the Codes of the Individual Elements</td>
<td>68</td>
</tr>
<tr>
<td>7.1</td>
<td>Overview of the individual software components</td>
<td>68</td>
</tr>
<tr>
<td>7.2</td>
<td>Recipe control (S7 control program)</td>
<td>73</td>
</tr>
<tr>
<td>7.3</td>
<td>Bridge Client (C# program)</td>
<td>80</td>
</tr>
<tr>
<td>7.4</td>
<td>Database structure (ACCESS / MySQL)</td>
<td>90</td>
</tr>
<tr>
<td>8</td>
<td>Adapting the Programs</td>
<td>101</td>
</tr>
<tr>
<td>8.1</td>
<td>Changes in the S7 program</td>
<td>95</td>
</tr>
<tr>
<td>8.2</td>
<td>Adapting the C# application</td>
<td>96</td>
</tr>
<tr>
<td>8.3</td>
<td>Creating a strong-typed DataSet</td>
<td>97</td>
</tr>
</tbody>
</table>
7 Explanation of the Codes of the Individual Elements

7.1 Overview of the individual software components

Recipe control (STEP 7 project with S7 control program)

The STEP 7 project contains the control program which provides the simulation of the I/O as well as the processing of the recipes.

Client Bridge

This Windows application developed in Visual Studio .NET 2003 communicates with WinAC and transmits new recipe data to the controller or reads the current values from the controller as required.

In addition, the application reads the recipe data from a database and writes the quality data of a recipe to a database.

ACCESS database

Based on MS Jet Engine, an ACCESS database can be accessed in Windows applications. An ACCESS file and the login data are transferred to the Jet Engine by connection string.

This enables a subsequent access in the Windows application as with conventional databases.

MySQL database

MySQL is a high-capacity database based on a database management system. The client/server architecture enables high performance also if the database is accessed by distributed stations.

If larger data amounts are to be managed and if many clients are involved, the use of a database with database management system should always be preferred.

7.2 Recipe control (S7 control program)

7.2.1 Program structures

Interaction of the blocks

The graphic below illustrates the interaction of the individual blocks. The data are basically transferred during block calls. In some exceptional cases, blocks access global data, which is also shown in the graphic for greater clarity.
7.2.2 Program flow

Program flow chart

Below, the S7 program is described, which provides the recipe processing in WinAC. Processing always starts in OB1.
### Table 7-1

<table>
<thead>
<tr>
<th>Flowchart</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OB1 cycle START</td>
<td>Start of organization block OB1.</td>
</tr>
<tr>
<td>FB_WorkingOfSteps Start</td>
<td>Transfer of the individual commands to FB_WorkingOfSteps and call of the block.</td>
</tr>
<tr>
<td>FB_Simulation</td>
<td>Transfer of I/O data to the simulation block and call of the block.</td>
</tr>
<tr>
<td>FB_DBIndex</td>
<td>Start of the data determination block and transfer of the results to block FB_WorkStep.</td>
</tr>
<tr>
<td>FB_WorkStep</td>
<td>Call of the block for step processing and transfer of the actual values.</td>
</tr>
<tr>
<td>FB_WorkingOfSteps End</td>
<td>End of the recipe processing block.</td>
</tr>
<tr>
<td>OB1 cycle END</td>
<td>End of organization block OB1.</td>
</tr>
</tbody>
</table>

### Program flowchart

For better understanding, the basic sequences in the program are displayed as program flowchart.
### Table 7-2

<table>
<thead>
<tr>
<th>Flowchart</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>START</strong></td>
<td>Start of the recipe processing (DB_Command_Glob.Start).</td>
</tr>
<tr>
<td>Send data request</td>
<td>Setting the DB_Command_Glob.DataRequest bit.</td>
</tr>
<tr>
<td>Have data been sent?</td>
<td>Checking the variable DB_Command_Glob.DataSend.</td>
</tr>
<tr>
<td>yes</td>
<td>Setting the variable DB_Command_Glob.DataSendAk.</td>
</tr>
<tr>
<td>no</td>
<td>The actual recipe processing starts. The number of steps is transmitted in DB_Command_Glob.</td>
</tr>
<tr>
<td>Sending the send acknowledgement</td>
<td>The data are read by calling FB_DBIndex.</td>
</tr>
<tr>
<td>Start recipe</td>
<td>Start of the processing by calling FB_Workstep.</td>
</tr>
<tr>
<td>Determining the number of worksteps</td>
<td>Using OB 1, the status information are written to DB_Visual.</td>
</tr>
<tr>
<td>Set variable $i = 0$</td>
<td></td>
</tr>
<tr>
<td>Reading the data for step $i$</td>
<td></td>
</tr>
<tr>
<td>Starting the processing of step $i$</td>
<td></td>
</tr>
<tr>
<td>Writing the status information</td>
<td></td>
</tr>
<tr>
<td>$i = i + 1$</td>
<td></td>
</tr>
<tr>
<td>Last step?</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Reseting the instances, and waiting for restart</td>
<td></td>
</tr>
</tbody>
</table>

### 7.2.3 Overview of function blocks and organization blocks

All blocks described in the following can be considered in STL or FBD, the comments in STL are more detailed.
OB1 OB_Main_Cycle

OB1 calls block FB_WorkingOfSteps with the values determined from the global data blocks and transfers the I/O data of FB_WorkStep to the global data block DB_Visual.

FB1 (FB_WorkStep)

This block is used to execute an individual workstep of the recipe. Execution starts with the transfer of the setpoints to the inputs and setting of the “Start” input. The graphic below illustrates the calling routine. Processing of the respective step of the recipe is started with the “Start” input. The block now performs the desired method. During the processing of the step, the Busy output is set to one. After completing the recipe step, the Ready bit for a PLC cycle is set and the Busy output is reset.

Parameters of block FB 1 (FB_WorkStep)

The data of the simulation block are transferred at the parameters with the prefix “Per...”. All parameters without this prefix are transfer points for the setpoints of the current step and are read and written by the block.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type/ data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingredient</td>
<td>IN/INT</td>
<td>Indicates the ingredient added in this workstep.</td>
</tr>
<tr>
<td>Method</td>
<td>IN/INT</td>
<td>Indicates the type of processing of the workstep.</td>
</tr>
<tr>
<td>Mix</td>
<td>IN/INT</td>
<td>Indicates the quantity of the ingredient to be added.</td>
</tr>
<tr>
<td>Reset</td>
<td>IN/BOOL</td>
<td>Reset (resets the block)</td>
</tr>
<tr>
<td>Speed</td>
<td>IN/INT</td>
<td>Indicates the speed of the stirring motor.</td>
</tr>
<tr>
<td>Start</td>
<td>IN/BOOL</td>
<td>This parameter starts the workstep.</td>
</tr>
<tr>
<td>Temperature</td>
<td>IN/INT</td>
<td>Indicates the temperature to be reached.</td>
</tr>
<tr>
<td>Unity</td>
<td>IN/INT</td>
<td>Defines the unit of measure of the ingredient to be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>added.</td>
</tr>
<tr>
<td>Busy</td>
<td>OUT/ BOOL</td>
<td>If this bit is set, the workstep is executed.</td>
</tr>
</tbody>
</table>
Part C: Program Description
Explanation of the Codes of the Individual Elements

Database Link to WinAC via OPC-XML
Entry ID: 21576581

<table>
<thead>
<tr>
<th>Name</th>
<th>Type / data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready</td>
<td>OUT/BOOL</td>
<td>After completion of the workstep, this output is set for a short time.</td>
</tr>
<tr>
<td>PerFillingLevel</td>
<td>IN/OUT INT</td>
<td>Indicates the current fill level of the tank.</td>
</tr>
<tr>
<td>PerTemperature</td>
<td>IN/OUT INT</td>
<td>Indicates the current temperature of the tank.</td>
</tr>
<tr>
<td>PerStirring-Speed</td>
<td>IN/OUT INT</td>
<td>Indicates the current speed of the mixer.</td>
</tr>
<tr>
<td>PerStirring</td>
<td>IN/OUT BOOL</td>
<td>Indicates whether the mixer is active.</td>
</tr>
<tr>
<td>PerHeating</td>
<td>IN/OUT BOOL</td>
<td>Indicates whether the tank heater is active.</td>
</tr>
<tr>
<td>PerCooling</td>
<td>IN/OUT BOOL</td>
<td>Signals operation of the tank cooler.</td>
</tr>
<tr>
<td>PerValve1, PerValve2, PerValve3</td>
<td>IN/OUT BOOL</td>
<td>Signals that the valves are open (2x inlet (1..2) and one outlet valve (3).</td>
</tr>
<tr>
<td>SP_Temp</td>
<td>IN/OUT INT</td>
<td>Indicates the temperature setpoint.</td>
</tr>
<tr>
<td>SP_Fill</td>
<td>IN/OUT INT</td>
<td>Indicates the quantity of the ingredient to be filled.</td>
</tr>
<tr>
<td>StepNr</td>
<td>IN/OUT INT</td>
<td>This parameter returns the currently processed workstep.</td>
</tr>
</tbody>
</table>

**FB 2 (FB_WorkingOfSteps)**

Function block FB2 is the core element of the recipe program. At the beginning it executes a step sequence managing the data request and acknowledgement after data reception. FB2 communicates with the Bridge Client. After receiving the data, it calls the function blocks FB_Simulation, FB_DBIndex and FB_WorkStep (see function description in the following sections).

It checks the processing of the individual worksteps and measures the time elapsed until the present stage of processing.

**Parameters of FB 2 (FB_WorkingOfSteps)**

Table 7-4

<table>
<thead>
<tr>
<th>Name</th>
<th>Type / data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecipeDBNr</td>
<td>IN / INT</td>
<td>Transfer of the number of the data block containing the recipes.</td>
</tr>
<tr>
<td>Reset</td>
<td>IN/BOOL</td>
<td>Resets all function blocks.</td>
</tr>
</tbody>
</table>
### Part C: Program Description

#### Explanation of the Codes of the Individual Elements

**Database Link to WinAC via OPC-XML**

**Entry ID:** 21576581

---

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequiredTime</td>
<td>OUT/INT</td>
<td>Returns the time required for the completed recipe in seconds.</td>
</tr>
<tr>
<td>CurrentStep</td>
<td>IN/OUT</td>
<td>Returns the currently processed step.</td>
</tr>
<tr>
<td>DataRequest</td>
<td>INT/OUT</td>
<td>Transfer of the recipe data request</td>
</tr>
<tr>
<td>DataRequestAk</td>
<td>IN/OUT</td>
<td>Transfer of the acknowledgement of the recipe data request.</td>
</tr>
<tr>
<td>DataSend</td>
<td>IN/OUT</td>
<td>Transfer of the send acknowledgement (from the Bridge Client).</td>
</tr>
<tr>
<td>DataSendAk</td>
<td>IN/OUT</td>
<td>Transfers an acknowledgement of reception (to the Bridge Client).</td>
</tr>
<tr>
<td>GoneTime</td>
<td>IN/OUT</td>
<td>Transfer of the time currently elapsed for the processed recipe.</td>
</tr>
<tr>
<td>MixProportion</td>
<td>IN/OUT</td>
<td>Transfer of the current fill level of the tank.</td>
</tr>
<tr>
<td>NrOfSteps</td>
<td>IN/OUT</td>
<td>Transfer of the number of worksteps of the current recipe.</td>
</tr>
<tr>
<td>PLCProgramRun</td>
<td>IN/OUT</td>
<td>Transfers the stage of recipe processing (1 recipe is being processed).</td>
</tr>
<tr>
<td>RecipeFinish</td>
<td>IN/OUT</td>
<td>Set after complete processing of the recipe.</td>
</tr>
<tr>
<td>Start</td>
<td>IN/OUT</td>
<td>Starts the processing of the recipe.</td>
</tr>
</tbody>
</table>

**FB3 (FB_DBIndex)**

Function block FB3 is used to determine the data of a workstep from the recipe data block DB_Recipe. The respective data are determined from the data block and returned to the calling block (in this application FB2) by transferring the number of the data block containing the recipe (in this application DB2000) and by indicating the workstep to be processed.

Viewing the block is only possible in STL.
Parameters of FB3 (FB_DBIndex)

Table 7-5

<table>
<thead>
<tr>
<th>Name</th>
<th>Type / data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NrGlobDB</td>
<td>IN/INT</td>
<td>Transfer of the number of the global data block containing the recipes.</td>
</tr>
<tr>
<td>IndexOfSet</td>
<td>IN/INT</td>
<td>Workstep to be processed in the recipe DB (in this application a max. number of 20 worksteps is possible).</td>
</tr>
<tr>
<td>Amount</td>
<td>OUT/INT</td>
<td>Returns the quantity of the ingredient to be filled into the tank.</td>
</tr>
<tr>
<td>Ingredient</td>
<td>OUT/INT</td>
<td>Returns the type of ingredient to be filled.</td>
</tr>
<tr>
<td>Proceed</td>
<td>OUT/INT</td>
<td>Indicates the method to be used for the specific workstep (e.g. stirring, heating, fill-in, etc.).</td>
</tr>
<tr>
<td>Speed</td>
<td>OUT/INT</td>
<td>Returns the speed of the mixer.</td>
</tr>
<tr>
<td>StepNr</td>
<td>OUT/INT</td>
<td>Returns the number of the selected workstep.</td>
</tr>
<tr>
<td>Temperature</td>
<td>OUT/INT</td>
<td>Returns the temperature to be reached.</td>
</tr>
<tr>
<td>Unity</td>
<td>OUT/INT</td>
<td>Indicates the unit of measure of the ingredient to be added.</td>
</tr>
</tbody>
</table>

FB225 (FB_Simulation)

Simulation block FB225 is used to simulate the I/O of the mixing and bottling plant. This block provides different input and output parameters. (Valves, heater switches, etc.)

If, for example, the heater input is set to 1, the temperature rises gradually.

Parameters of FB225 (FB_Simulation)

Table 7-6

<table>
<thead>
<tr>
<th>Name</th>
<th>Type / data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartFlag</td>
<td>IN/BOOL</td>
<td>If this input is set, the simulation block is active.</td>
</tr>
<tr>
<td>Reset</td>
<td>IN/BOOL</td>
<td>If this input is set, the block is reset.</td>
</tr>
<tr>
<td>TimerFlag</td>
<td>OUT/BOOL</td>
<td>Provides a flag with a trigger time of 0.5 sec.</td>
</tr>
<tr>
<td>cooling</td>
<td>IN/OUT BOOL</td>
<td>Reading and writing the cooler status.</td>
</tr>
<tr>
<td>fillinglevel</td>
<td>IN/OUT INT</td>
<td>Returns the fill level in the tank.</td>
</tr>
<tr>
<td>heating</td>
<td>IN/OUT INT</td>
<td>Reading and writing the heater status.</td>
</tr>
</tbody>
</table>
7.2.4 Overview of the data blocks

**DB2000 (DB_Recipe)**

As buffer for recipes, DB2000 is the most important data block. The complete recipe to be processed is stored in this block.

DB2000 consists of 20 individual steps which are respectively created from the UDT1 (UDT_Recipe) data structure.

Data of DB2000 (DB_Recipe)

The data of the recipe data block consist of 20 UDT1 structures. This results in the following parameters for step 1. In the client, this block is accessed directly.

Except for the step number, all parameters listed in the following are equivalent.
### Table 7-7

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step1.StepNr</td>
<td>INT</td>
<td>Contains the number of the step to ensure correct identification.</td>
</tr>
<tr>
<td>Step1.Procceed</td>
<td>INT</td>
<td>Contains the ID of the processing method for the step (heating, cooling, fill-in, etc.).</td>
</tr>
<tr>
<td>Step1.Ingredient</td>
<td>INT</td>
<td>Indicates the ingredient to be used.</td>
</tr>
<tr>
<td>Step1.Amount</td>
<td>INT</td>
<td>Stores the quantity of the required ingredient.</td>
</tr>
<tr>
<td>Step1.Unity</td>
<td>INT</td>
<td>Indicates the unit for the required ingredient.</td>
</tr>
<tr>
<td>Step1.Temperature</td>
<td>INT</td>
<td>Indicates the temperature to be reached.</td>
</tr>
<tr>
<td>Step1.Speed</td>
<td>INT</td>
<td>Indicates the stirring speed of the mixer.</td>
</tr>
</tbody>
</table>

### DB3000 (DB_Command_Glob)

DB3000 was integrated into the recipe program for the exchange of commands and status information between controller and Bridge Client. In the Bridge Client, the block is accessed directly.

### Data of DB3000(DB_Command_Glob)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>BOOL</td>
<td>This bit starts the recipe.</td>
</tr>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>This bit resets all blocks.</td>
</tr>
<tr>
<td>DataRequest</td>
<td>BOOL</td>
<td>The DataRequest bit is set by the controller to request recipe data from the Bridge Client.</td>
</tr>
<tr>
<td>DataRequestAk</td>
<td>BOOL</td>
<td>The Bridge Client sets this bit to acknowledge the data request.</td>
</tr>
<tr>
<td>DataSend</td>
<td>BOOL</td>
<td>DataSend is sent by the Bridge Client if the data have been transmitted.</td>
</tr>
<tr>
<td>DataSendAk</td>
<td>BOOL</td>
<td>With DataSendAk, the controller sends a bit to acknowledge the reception of the data.</td>
</tr>
<tr>
<td>PLCProgramRun</td>
<td>BOOL</td>
<td>With this bit, the Bridge Client is informed on recipe processing by the controller.</td>
</tr>
<tr>
<td>RecipeFinish</td>
<td>BOOL</td>
<td>After completion of the recipe, this bit is set by the controller.</td>
</tr>
<tr>
<td>RequiredTime</td>
<td>INT</td>
<td>Indicates the required time in seconds.</td>
</tr>
<tr>
<td>CurrentStep</td>
<td>INT</td>
<td>CurrentStep is an integer value which indicates the step currently processed by the controller.</td>
</tr>
<tr>
<td>NrOfSteps</td>
<td>INT</td>
<td>Transfer of the total number of steps in the recipe.</td>
</tr>
<tr>
<td>MixProportion</td>
<td>INT</td>
<td>Indicates the current tank fill level for analysis.</td>
</tr>
</tbody>
</table>
**Part C: Program Description**

**Explanation of the Codes of the Individual Elements**

*Database Link to WinAC via OPC-XML*

**GoneTime**

DINT Indicates the time currently required for the recipe.

---

**DB4000 (DB_Visual)**

The Bridge Client features an integrated visualization. The visualization is started only if required to avoid unnecessary communication loads. For the transfer of the actual values from the controller to the Bridge Client, a separate visualization data block was generated, which stores the actual values of the recipe processing in real time (updated in each cycle run).

**Data of DB4000(DB_Visual)**

The variables of DB4000 are parameter data of the function block FB_WorkStep. Aside from the prefix “Per”, the names of the variables are identical. Since the meaning and the data types are identical, this section does not include a detailed description (see section Parameters of FB1 (FB_WorkStep)).

---

**7.2.5 Description of selected code excerpts**

**Code excerpt of the data request of the S7 program**

The following code excerpt from FB2 (WorkingOfSteps) shows step 1 of the step sequence which requests the recipe data from the client after starting the recipe processing.

*Netzwerk 1: Datenanforderungsschritt (Schritt1) / Data request step (step 1)*

U Start // Startbit gesetzt? / If the StartFlag is set
S #FStartData // setze FStartdata (Schritt1) / then set FStartData (Step 1) Flag
O #FdataRequestAk // Die Rücksetzung mit nächstern Schritt, / Resetting by Step 2 (FDataRequestAk)
O #Reset // oder Reset Bit / or reset flag.
R #FStartData
U #FStartData
= #DataRequest // Setze DataRequest auf 1, wenn Schritt 1 aktiv / set DataRequest if the Step 1 is active
R #RecipeFinisch // rücksetzen des RecipeFinisch Bits / reset RecipeFinisch flag

**Description of the data request step**

The OPC variables (transmitted with global DB_Command_Glob) are transferred to FB2 (FB_WorkingOfSteps) in OB1. The names and values of the four variables “Start”, “DataRequest”, “Reset” and “RecipeFinish” are identical to the names and data stored in the global DB.
As soon as the value is set in the Start bit, the network activates FdataRequestAk and returns the data request to the OPC server with the DataRequest bit. The "RecipeFinish" status is reset simultaneously.

Code excerpt of the processing of a recipe workstep in the S7 program

The following code excerpt shows a call of FB1 (FB_WorkStep) in function block FB2 (FB_WorkingOfSteps) in network 6.

Netzwerk 6: Aufruf des Einzelarbeitsschrittes / Call the single work step

CALL "FB_WorkStep", "DB_WorkStep"
Start  := #StartWorkStep // Starting by setting the StartWorkStep flag
Ingredient :="DB_FB_DBIndex".Ingredient    // Übergabe der verwendeten Zutat
Mix      :="DB_FB_DBIndex".Amount     // Übergabe der verwendeten Menge
Unity    :="DB_FB_DBIndex".Unity     // Übergabe der Einheit der Zutat
Speed    :="DB_FB_DBIndex".Speed      // Übergabe des Sollw. der Rührgeschw.
Method   :="DB_FB_DBIndex".Proceed    // Übergabe der gewünschten Methode
Reset    :=#Reset
Busy      :=
Ready     :=
StepNr    :="DB_FB_DBIndex".StepNr    //Übergabe des aktuellen Schrittes
SP_Temp   :=
SP_Fill   :=
PerFillingLevel :="DB_Simulation".fillinglevel // Übergabe des akt. Füllst.
PerTemperature :="DB_Simulation".temperature //Übergabe der akt. Temp.
PerStirringSpeed :="DB_Simulation".stirringspeed // Übergabe der akt. Rührgeschw.
PerStirring :="DB_Simulation".stirring    // Übergabe des akt. Rührer Zust.
Heizung / transfer current state of heating

\[
\text{PerCooling} \ := \ "DB\_Simulation"\text{.cooling} \ // \ \text{Überg. des akt. Kühlungs Zust.}
\]
// handing over the current state of cooler

\[
\text{PerValve1} \ := \ "DB\_Simulation"\text{.valve1} \ // \ \text{Übergabe des akt. Ventil 1 Zust.}
\]
// handing over the current state of valve 1

\[
\text{PerValve2} \ := \ "DB\_Simulation"\text{.valve2} \ // \ \text{Übergabe des akt. Ventil 2 Zust.}
\]
// handing over the current state of valve 2

\[
\text{PerValve3} \ := \ "DB\_Simulation"\text{.valve3} \ // \ \text{Übergabe des akt. Ventil 3 Zust.}
\]
// handing over the current state of valve 3

\[
\ldots
\]

**Description of the call of the workstep processing**

The network displayed above calls block FB 1 (FB_WorkStep) and transfers the required setpoints from the instance data block of the index function block FB 3 (FB_DBIndex). In addition, the current values (with the prefix Per…) are written to the data block by the instance data block of the simulation block (FB_Simulation) and read (bi-directional). The processing is started via the local variable #StartWorkstep. Resetting is performed using the local variable #Reset.

**7.3 Bridge Client (C# program)**

**7.3.1 Principles of operation of the client**

**Overview**

The client was developed for .NET Framework in the C# programming language.

It combines 3 basic functionalities exchanging data on one interface. These functionalities are OPC-XML access, database accesses using ADO.NET and visualization of the recipe processing. The following chapters provide a detailed description of these three functionalities.

Basically, the client consists of 2 form classes, the basic functionality (data exchange between database and WinAC) being provided by the “basicForm”. The form class “VisualForm” provides the visualization (documented briefly).

For better understanding, the encapsulation of the functionalities in separate classes will not be described in the following.

**Flowchart**

The flowchart below illustrates the basic principle of operation of the automatic mode.
Structure of method calling (simplified)

The following sequence structure, which displays the interaction of the core methods, provides a basic overview of the specific sequence of the program. “Important methods” provides an overview of the most important methods.

Figure 7-5

```
<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start_Client_Click</td>
<td>This button click method executes the following steps to start the client in automatic mode.</td>
</tr>
<tr>
<td></td>
<td>Step 1: Connecting to the database (ConnectToMySQLDB / ConnectToACCESSDB)</td>
</tr>
<tr>
<td></td>
<td>Step 2: Connecting to the OPC server (ConnectToOPCXML)</td>
</tr>
<tr>
<td></td>
<td>Step 3: Registering subscription (RegisterSubscription)</td>
</tr>
<tr>
<td></td>
<td>Step 4: Starting the timer for polling</td>
</tr>
<tr>
<td>ConnectToMySQLDB</td>
<td>This method establishes a connection to the MySQL database and writes the data of the tables “Expiry”, “Protocol” and “Quality” to the “DataSetFull” DataSet</td>
</tr>
</tbody>
</table>
```
### ConnectToACCESSDB
This method establishes a connection to the ACCESS database and writes the data of the tables “Expire”, “Protocol” and “Quality” to the “DataSetFull” DataSet.

### ConnectToOPCXML
This method tests the web service and the status of the OPC server and returns a status dialog window.

### RegisterSubscription
RegisterSubscription signals the data of the command variables to be polled to the OPC server.

### timer1_Tick
After the timer has elapsed, the call of the subscription request is sent to the server.

### SubScrCallback
If the server identifies a change of the registered data, this method is called in the client. Using this method, the changed value is identified and a corresponding reaction is defined. The three most important value changes are:
- Data requested?
  - a. TransferRecipe
- Current step changed?
  - b. E1_StepChanged
- Current recipe completed?
  - c. E2_Finisch

### E1_StepChanged
This event writes the data (runtime, time stamp, fill level, etc.) of the currently processed individual workstep into the “Protocol” table of DataSetFull.

### E2_Finisch
This event writes the data (runtime, time stamp, etc.) of the currently processed recipe into the “Quality” table of DataSetFull.

## 7.3.2 OPC-XML access (C# Windows program)

### Basics
As described in /2/, the OPC server is accessed with subscriptions and synchronous calls.

### Transfer of the recipe data to the controller
The recipe data are transferred with a synchronous OPC-XML write command. The TransferRecipe method transfers a recipe to the controller. The following code excerpt shows the most important routines of this method.
Code excerpt

```csharp
private void TransferRecipe(int NrOfRecipe)
{
    // Check the number of steps and position in the table
    int PosRecipe;
    int NrOfSteps=HowMuchSteps(NrOfRecipe,DataSetFull,out PosRecipe);
    // Generate the writelist for write method
    // Setup the items of the list
    WDataItemList.Items = new ItemValue[(NrOfSteps*7)+1];
    for(int i=0;i<((NrOfSteps*7)+1);i++)
    {
        WDataItemList.Items[i] = new ItemValue();
    }
    // Assigning the itemnames of the items in the list
    // Assigning dynamically for better performance
    // i represents the step number of write list and j represents the row in the database
    int j=0;
    string js="";
    for (int i=0;i<NrOfSteps*7;i=i+7)
    {
        j++;
        js=Convert.ToString(j);
        // Assigning the step name "PCWinAC.WinLC RTX.DB_Recipe.Step j .StepNr" for step j
        WDataItemList.Items[i].ItemName=
            cstation+"."+cPLC+"."+cRecipeDB+"."+cUDT+js+"."+cINStepNr;
        // and so on (for the next 6 values)
    }
    // Adding the name NumberOfSteps for the DB_Command_Glob
    // "PCWinAC.WinLC RTX.DB_Command_Glob.NrOfSteps"
    WDataItemList.Items[(NrOfSteps*7)].ItemName=
        cstation+"."+cPLC+"."+cCommandDB+"."+cINNrOfSteps;
    j=0;
    // write database values to writelist
    for (int i=0;i<(NrOfSteps*7);i=i+7)
    {
        js=Convert.ToString(j);
        // transfer the database value of “OrderNumber” from database to "StepNumber" of writelist
        WDataItemList.Items[i].Value =DataSetFull.Expiry[j+PosRecipe].OrderNumber;
        WDataItemList.Items[i+1].Value=DataSetFull.Expiry[j+PosRecipe].Method;
        // and so on (for the next 6 values)
        j++;
    }
    // Write the item list to OPC Server
    WriteItemList(ref WDataItemList);
}
```
Explanation of the TransferRecipe method

The desired recipe number is transferred to the method. At the beginning of the method call, the number of steps of the respective recipe and the position in the Expiry table are determined. Subsequently, the client generates a list of write items for the transfer of the individual data depending on the number of steps. In the next step, the individual values of the recipe are transferred from the DataSet to the list of write items. Finally, the WriteItemList method is called, which transfers the list to the OPC server (or the controller).

7.3.3 Database accesses (C# Windows program)

Basics

As shown in Chapter 4.5 Basics of the ADO.NET interface, the .NET DataAdapter provides a convenient option to access a database. This requires the .NET providers for the individual database types. In this application, the two providers Microsoft Jet 4.0 OLE DB Provider for an ACCESS database and ByteFx .NET DataProvider for access to a MySQL database are used.

Microsoft Jet 4.0 OLE DB Provider (for link to ACCESS)

MS Jet 4.0 OLE DB Provider is already integrated in Visual Studio and can be activated with the following two instructions:

```csharp
using System.Data;
using System.Data.OleDb;
```

Byte FX .NET DataProvider (for link to MySQL)

Byte FX Provider is a library with the interface of a DataProvider for .NET and can be used freely. Its installation has already been described in Chapter 5.1 Preparatory installation.

The instructions below show how Byte FX Provider can be integrated into a Visual Studio project.
Integrating Byte FX .NET DataProvider into a VS 2003 project

Table 7-10

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If not yet done, open Visual Studio. <strong>Open</strong> the project into which you want to integrate the library or create a new project.</td>
</tr>
<tr>
<td>2</td>
<td>Click <strong>“References”</strong> in the Project Explorer and select <strong>“Add References”</strong>.</td>
</tr>
<tr>
<td>3</td>
<td><strong>Verweis hinzufügen</strong></td>
</tr>
</tbody>
</table>

A dialog box appears with a **“.NET”** tab in which the two entries **“ByteFX.MySqlClient.Design”** and **“ByteFX.MySQLClient.dll”** should be displayed.

**Note:** If these entries are not displayed, click **“Browse...”** and add the two DLL libraries **“ByteFX.MySqlClient.Design.dll”** and **“ByteFX.MySQLClient.dll”** from the installation path of the libraries.

<table>
<thead>
<tr>
<th>Komponentenname</th>
<th>Version</th>
<th>Pfad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessibility.dll</td>
<td>1.0.5000.0</td>
<td>C:\WINNT\Microsoft\Net\Fra...</td>
</tr>
<tr>
<td>adodb</td>
<td>7.0.3300.0</td>
<td>C:\Program Files\Microsoft.Net...</td>
</tr>
<tr>
<td>ByteFX.MySqlClient.Design</td>
<td>0.7.6.15078</td>
<td>D:\Program Files\ByteFX\Byte...</td>
</tr>
<tr>
<td>ByteFX.MySqlClient.dll</td>
<td>0.7.6.15073</td>
<td>D:\Program Files\ByteFX\Byte...</td>
</tr>
<tr>
<td>CrystalDecisions.CrystalReportDesigner.dll</td>
<td>9.1.5000.0</td>
<td>C:\Program Files\Common File...</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Komponentenname</td>
<td>Version</td>
<td>Pfad</td>
</tr>
<tr>
<td>Accessibility.dll</td>
<td>1.0.5000.0</td>
<td>C:\WINNT\Microsoft\Net\Fra...</td>
</tr>
<tr>
<td>adodb</td>
<td>7.0.3300.0</td>
<td>C:\Program Files\Microsoft.Net...</td>
</tr>
<tr>
<td>ByteFX.MySqlClient.Design</td>
<td>0.7.6.15078</td>
<td>D:\Program Files\ByteFX\Byte...</td>
</tr>
<tr>
<td>ByteFX.MySqlClient.dll</td>
<td>0.7.6.15073</td>
<td>D:\Program Files\ByteFX\Byte...</td>
</tr>
<tr>
<td>CrystalDecisions.CrystalReportDesigner.dll</td>
<td>9.1.5000.0</td>
<td>C:\Program Files\Common File...</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

4 Select **“ByteFX.MySqlClient.Design”** and click the **“Select”** button; proceed identically for **“ByteFX.MySQLClient.dll”** and subsequently click **“OK”**.

5 Now enter the following definitions before the start of the namespace of your project to integrate the library into the definition part.

```csharp
using ByteFX;
using ByteFX.Data;
using ByteFX.Data.MySqlClient;
using ByteFX.Data.MySqlClient.Design;
```

The integration is now completed.
Creating a Dataset-based database link (ACCESS)

The following section describes how a DataGrid (data table) polling an ACCESS database can be created using the OleDbDataAdapter component. You require a Microsoft Visual Studio .NET 2003 development environment and the RecipeAdministration.mdb file included in the delivery.

Table 7-11

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If not already done, open Visual Studio and click “File-&gt;New -&gt;Project”. Select the project types “Visual C# projects” and the “Windows Application” template. Select the storage location and name for your project and create the project.</td>
</tr>
<tr>
<td>2</td>
<td>Click the “Data” tab in the toolbox and move “OleDbDataAdapter” to your form using drag-and-drop.</td>
</tr>
</tbody>
</table>
| 3   | The DataAdapter configuration wizard opens.  
  - Click “Next”.  
  - Click the “New Connection” button. |
| 4   | Click the “Provider” tab and subsequently select “Microsoft Jet 4.0 OLE DB Provider”.  
  Click the “Next” button. |
| 5   | Specify the following to connect to Access data:  
  1. Select or enter a database name:  
  Now click the “Browse...” button to find the ACCESS file (RecipeAdministration.mdb) and your path.  
  Set the password specifications (no password is used in the ACCESS database) and click “OK”. |
6. Follow the instructions of the wizard and select “Use SQL statements” in the dialog box for query of the polling type. Follow the further instructions.

7. Click “Query Builder” in the “Generate SQL-Statements” dialog box.

- Add the “Expiry” table and close the “Add Table” dialog box.
- Check “*(AllColumns)” in the “Expiry” table and click “OK” to close the query builder.

8. The wizard now displays the automatically generated SQL statement. Follow the instructions of the wizard and click “Finish”. 
With the wizard, a new DataAdapter and a new connection to an OLE database have been created.

Click **“oleDbDataAdapter”** with the right mouse button and select **“Generate DataSet”**.

Visual Studio now generates an XML file named **“DataSet1”**, which includes the structure of the selected data as XML structure. In addition, a C# file named **“DataSet1.CS”** is generated, which includes the typed structures in the form of the DataSet1 class.

As instance of the **“DataSet1”** class, the **“dataSet1 1”** component visible in the form is generated.

In **“Windows Forms”**, now drag **“DataGridView”** to your form and adapt the size in such a way that the form is not completely covered.

- Click DataGridView with the right mouse button and select **“Properties”**.
- Select **“dataSet11”** as **“DataSource”** and **“Expiry”** as **“DataMember”**.

Now add **Button** using drag and drop.

Double-click this button and insert the following code line into the shown method:

```csharp
oleDbDataAdapter1.Fill(dataSet11);
```
Part C: Program Description
Explanation of the Codes of the Individual Elements

Database Link to WinAC via OPC-XML

Press the "F5" to start the application.
As soon as the form is displayed, you can click the self-created button and the DataGrid is filled with the data from the table.

This short introduction illustrates how quickly typed DataSets can be created and used. The creation of typed DataSets with several tables as used in this application is explained in Chapter 8.3 Creating a strong-typed DataSet.

Equivalence of MySQL and ACCESS
Due to the use of .NET data adapters, the source code for data exchange with the two databases is almost identical. In each case only one DataSet is used for both databases. Differences in data processing refer only to the connection to the database, the filling of the DataSet and the database updating (return of the data).

The following code examples refer to the use of an ACCESS database. The source code for a MySQL processing differs only insignificantly from the code of the following example and is commented in detail in the project included in the delivery.

Code excerpt of the connection to the database (Connection)

```csharp
    string connectionString;
    // generate the connection string
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\RecipeAdministration.mdb;USER ID=;PASSWORD=;"
    // generate the OleDbConnection1
    OleDbConnection1 = new OleDbConnection(connectionString);
    // open the connection
    OleDbConnection1.Open();
```
Explanation of the code excerpt – connecting to the database

Before processing data of a database, it is first required to establish a connection to the database. A ConnectionString transferring all connection information to the DataProvider is always used for this connection.

However, the ConnectionString is different for each database type.

The ACCESS ConnectionString transfers only the position of the ACCESS file (path and file name *.mdb) and the user information.

MySQL Byte FX .NET Provider, however, requires a ConnectionString with the following information:

```java
myConnString = "Persist Security Info=False; database=Recipe;
server=localhost;Connect Timeout=30;
user id=Admin;pwd=";
```

The MySQL ConnectionString transfers information on security, the server (name and place) and database designation.

After the generation of the ConnectionString, a new connection is established and opened.

Code excerpt – generating the DataAdapters

```java
// generate a new OleDb DataAdapter for Expiry table with SQL statement
// and the open connection
oleDbDataAdapterExpiry=new OleDbDataAdapter("SELECT * FROM Expiry ORDER BY ID",oleDbConnection1);
// generate a new OleDb DataAdapter for the Protocol table with SQL
// statement and the open connection
oleDbDataAdapterProtocol=new OleDbDataAdapter("SELECT * FROM protocol",oleDbConnection1);
// generate a new OleDb DataAdapter for the Quality table with SQL
// statement and the open connection
oleDbDataAdapterQuality=new OleDbDataAdapter("SELECT * FROM quality",oleDbConnection1);
```

Explanation of the generation of the DataAdapters

To enable easy updating of the database, a separate DataAdapter has been generated for each table. These DataAdapters are generated when an SQL string is transferred and the connection described above is specified.

Code excerpt – filling the DataSet

```java
// declare DataSetFull as the main data handle object
private NewDataSet DataSetFull;
// fill the dataset table Expiry with the table data of the
// OleDbDataAdapterExpiry
oleDbDataAdapterExpiry.Fill(DataSetFull,"expiry");
// fill the dataset table Protocol with the table data of the
// OleDbDataAdapterProtocol
oleDbDataAdapterProtocol.Fill(DataSetFull,"protocol");
// fill the dataset table Quality with the table data of the
// OleDbDataAdapterQuality
```
oleDbDataAdapterQuality.Fill(DataSetFull,"quality");

Explanation of filling the DataSet

To fill the DataSet with the data of the DataAdapter, the DataSet is transferred as parameter in the DataAdapter.Fill method. In addition, an ID (in most cases the table name) is transferred to the method for identification. All tables are filled into the same DataSet.

Code excerpt – Updating the database

```csharp
// generate a new OleDb command handler for the Expiry table
oleDBCommandBuilderExpiry = new OleDbCommandBuilder(oleDbDataAdapterExpiry);

// generate a new OleDb command handler for the Protocol table
oleDBCommandBuilderProtocol = new OleDbCommandBuilder(oleDbDataAdapterProtocol);

// generate a new OleDb command handler for the Quality table
oleDBCommandBuilderQuality = new OleDbCommandBuilder(oleDbDataAdapterQuality);

// update the database with the help of oleDbDataAdapter
oleDbDataAdapterExpiry.Update(DataSetFull,"Expiry");
oleDbDataAdapterProtocol.Update(DataSetFull,"Protocol");
oleDbDataAdapterQuality.Update(DataSetFull,"Quality");
```

Explanation of updating the database

After processing the individual data within the DataSet (DataSetFull), the data can be written back to the database with the Update method of the corresponding DataAdapter. With this Update method, the SQL commands for table processing are automatically sent to the database.

Precondition for the use of the Update method is the creation of a CommandBuilder which generates the commands for the respective DataAdapter.

Note

The data adapters are updated via Update using optimistic locking. For this reason, you should implement a comprehensive error control in the application if the database is simultaneously accessed by several users.

7.3.4 Visualization (C# Windows program)

The program code of the visualization of the Bridge Client is basically similar to the code described in /2/.
7.4 Database structure (ACCESS / MySQL)

Overview

The two databases consist of a total of 11 tables; three of these tables will be described in detail in the following sections:

1. Expiry table (contains the recipes)
2. Protocol (logging) table
3. Quality table

7.4.1 Tables

Expiry table

The figure below shows a section of the used Expiry table, which includes all information of the recipe.

![Figure 7-6](image)

<table>
<thead>
<tr>
<th>ID</th>
<th>NameStep</th>
<th>Recipe</th>
<th>OrderNumber</th>
<th>Method</th>
<th>Ingredient</th>
<th>Amount</th>
<th>Unit</th>
<th>Temp</th>
<th>SPEED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>admission of water (apple juice)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>150</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>heating on chafing dish (apple juice)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>100</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>pouring of apple juice (apple juice)</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>200</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>pouring of sugar (syrup) 2 (apple juice)</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>3000</td>
</tr>
<tr>
<td>5</td>
<td>cooling down (filling 3 (apple juice)</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>100</td>
<td>0</td>
<td>100</td>
<td>25</td>
<td>4000</td>
</tr>
</tbody>
</table>

Explanation of the Expiry table

The Expiry table is the most important table in the recipe database. It contains the process information necessary for the recipe and each row corresponds to an individual step of a recipe. The most important columns are the Method column, which specifies the method to be applied, the Ingredient column, the Amount column and the Temp column (temperature to be reached).

Protocol table

The workstep data are logged with the Protocol table shown below:

![Figure 7-7](image)

<table>
<thead>
<tr>
<th>ID</th>
<th>RecipeNumber</th>
<th>StepNumber</th>
<th>GoneTime</th>
<th>MixProportion</th>
<th>DTSStamp</th>
<th>Station</th>
<th>PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>385</td>
<td>0</td>
<td>2004-06-19 14:58:13</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2004-06-19 14:50:18</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>76</td>
<td>150</td>
<td>2004-06-19 14:57:32</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>88</td>
<td>150</td>
<td>2004-06-19 14:57:44</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Explanation of the Protocol table

After each completed workstep, the Protocol table stores the number of the currently processed recipe (RecipeNumber), the finished workstep (StepNumber), the time elapsed until the current stage of processing (GoneTime), a time stamp and information on the respective controller (station, PLC). However, the data are only written to the database after completion of a recipe.
Quality table

The Quality table is used to store information on the completed recipes.

<table>
<thead>
<tr>
<th>ID</th>
<th>RecipeNumber</th>
<th>RequiredTime</th>
<th>DTStamp</th>
<th>Station</th>
<th>PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>1</td>
<td>408</td>
<td>2004-06-14 22:32:06</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>408</td>
<td>2004-06-14 22:48:10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>0</td>
<td>2004-06-15 09:05:32</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>557</td>
<td>2004-06-15 09:14:59</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Explanation of the Quality table

The Quality table is used to store information on the completed recipes. This includes storing the respective recipe number (RecipeNumber), the time required for the recipe (RequiredTime), a time stamp and information on the controller. Each row of the table corresponds to a completed recipe.
8 Adapting the Programs

8.1 Changes in the S7 program

This example

The S7 program included in the delivery of this application includes only four important features:

- Data exchange with the OPC server (or WinAC)
- Data exchange with an ACCESS / MySQL database
- Data editing for both directions
- Visualization of important data.

Adaptation for larger recipes

The maximum number of individual recipe steps is currently limited to 20. Since this number refers to a predefined global data block, it is also additionally required at least in the main memory of WinAC. It would be more useful to dynamically adapt the data block depending on the used individual steps.

The data block can be easily extended by creating the desired number of steps as UDT structure (UDT_Recipe) with the name Stepxx in data block DB2000 (DB_Recipe) in addition to the existing steps.

Adaptation for real I/O

The I/O is only simulated in this application. The use of external I/O requires to use a SIMATIC ET 200 station with analog and digital inputs and outputs.

In this case, please remove the simulation block FB_Simulation in the S7 program.
Then link the I/O data (PerFillingLevel, PerValve1, etc.), which are transferred to FB_WorkStep in FB_WorkingOfSteps (network 6), to the real inputs.

In addition, it will certainly be required to standardize the values. A function block can be used for standardization, which can basically be used the same way as the simulation block (direct transfer of instance data).

8.2 Adapting the C# application

Error control when writing a DataSet to a database (updating)

As already mentioned above, the processing of table data with DataSets is very convenient. The only exception is the automatic update function with the aid of the DataAdapter.

If the data in the database are changed during processing in the DataSet, an error message will possibly be displayed during automatic updating.

In order to correct this error, you have to check which value is affected. This value is then again read from the database and the DataSet is adapted accordingly.

Now it should be possible to write the complete DataSet to the database.

Modifying the database properties in the C# program

If you want to adapt the database properties to your requirements, the typed DataSet must be regenerated (see 8.3 Creating a strong-typed DataSet) and additionally the “TransferRecipe” method has to be modified.
If you want to change the table names, further extensive modifications are required which refer to the individual methods `ConnectToACCESSDB`, `ConnectToMySQL`, `E1_StepChanged`, `E2_Finish`, `WriteFinishToDataSet`, `WriteStepToDataSet`, `HowMuchSteps`, `CheckRecipe` and the `btnUpdate_Click` method.

Changing the station and controller name

If you want to change the name of the station or of the controller in the Step 7 project, the following lines in the declaration part of the C# program have to be adapted:

```csharp
private const string cstation = "PCWinAC";
private const string cPLC = "WinLC RTX";
```

Adapt these settings in the source file “`BasicForm.CS`” and in the source file “`VisualForm.CS`”.

8.3 Creating a strong-typed DataSet

Problem

In the Visual Studio 2003 .NET developing environment, it is not possible to create a typed DataSet with several tables using the wizard available for the ACCESS (or MySQL) database. In the C# application, such a DataSet was used. The following instructions show you how to create such a DataSet with the aid of a small C# project.

This procedure is required if you want to adapt the C# program, if you want to use another database or if you want to change the table properties (names of columns, tables, etc.).

This procedure is not necessary if you only want to change the contents of the tables.

Solution

A strong-typed DataSet is a derivation of the DataSet class which includes the names of the tables and respective columns as properties of the “NewDataSet” class. This derivation can be generated with the xsd.exe program (located in the directory `C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`) from an XML file which includes the data structure. In turn, this XML file is generated from a non-typed DataSet with the “DataSetx.FillSchema” method.

Implementation

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open Visual Studio .NET 2003 and create a new C# project.</td>
</tr>
<tr>
<td>2</td>
<td>Insert the code excerpt from Chapter 7.3.3 Database accesses (C# Windows program) under “Code excerpt of the connection to the database (Connection)” into your project and change the code if required (source of the database, etc.).</td>
</tr>
</tbody>
</table>
3. Now generate 3 DataAdapters with the following code.

```csharp
oleDbDataAdapter1 = new OleDbDataAdapter("SELECT * FROM Expiry", oleDbConnection1);
oleDbDataAdapter2 = new OleDbDataAdapter("SELECT RecipeNumber, RequiredTime, DTStamp, Station, PLC FROM Quality", oleDbConnection1);
oleDbDataAdapter3 = new OleDbDataAdapter("SELECT NumberOfRecipe, StepNumber, GoneTime, MixProportion, DTStamp, Station, PLC FROM Protocol", oleDbConnection1);
```

Note: Please do not use * as column name in the SQL string of the tables Protocol and Quality, since otherwise the ID which must not be used is inserted.

4. Now fill the adapter using the “FillSchema” method. The name in quotation marks is the name later available in the DataSet.

```csharp
oleDbDataAdapter1.FillSchema(dataSet1, SchemaType.Mapped, "Expiry");
oleDbDataAdapter2.FillSchema(dataSet1, SchemaType.Mapped, "Quality");
oleDbDataAdapter3.FillSchema(dataSet1, SchemaType.Mapped, "Protocol");
```

5. The “WriteXmlSchema” method of the DataSet now generates an XML file at the defined position.

```csharp
dataSet1.WriteXmlSchema("C:\DataSetFull.xsd");
```

6. Copy this file to the directory “C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin”

Note: The directory may vary depending on the specific installation.

7. Open the xsd.exe file with the following parameters.

   ```csharp
   xsd.exe /d /l:C# DataSetFull.xsd
   ```

   Note: The parameters can be called with a link as described in Chapter 6.1 Operating the MySQL database.

8. Please copy the newly created `DataSetFull.cs` file to the directory of the project in which you want to insert / change the typed access to the database.

   Now integrate the generated class into the Project Folder Editor by clicking the right mouse button and selecting “References>Add References...”.

   ```csharp
   oleDbDataAdapter1.FillSchema(dataSet1, SchemaType.Mapped, "Expiry");
   oleDbDataAdapter2.FillSchema(dataSet1, SchemaType.Mapped, "Quality");
   oleDbDataAdapter3.FillSchema(dataSet1, SchemaType.Mapped, "Protocol");
   ```

   The “WriteXmlSchema” method of the DataSet now generates an XML file at the defined position.

   ```csharp
dataSet1.WriteXmlSchema("C:\DataSetFull.xsd");
```
9  With the command

```csharp
private NewDataSet DataSetFull;
```

you can instance the generated NewDataSet class as DataSetFull and use it with typed calls from this time on.
**Part D: Appendix**

### 9 Bibliographic References

**Bibliographic references**

This list is by no means complete and only provides a selection of appropriate sources.

Further literature is also available at A&D Service & Support. A&D Service & Support is available on the internet at: [http://support.automation.siemens.com](http://support.automation.siemens.com) (Enter the Entry ID in the search field).

---

#### Table 9-1

<table>
<thead>
<tr>
<th>No.</th>
<th>Topic</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>/1/</td>
<td>Application example on PC-based automation with SIMATIC WinAC A&amp;D Service &amp; Support</td>
<td>PC-Based Automation: Basics for the Solution of Automation Tasks Based on WinAC RTX Entry ID: <a href="#">21004765</a></td>
</tr>
<tr>
<td>/2/</td>
<td>Application example on PC-based automation with SIMATIC WinAC A&amp;D Service &amp; Support</td>
<td>PC-Based Automation: Plant Visualization with Visual Basic .NET via the OPC XML-DA Interface Entry ID: <a href="#">21004994</a></td>
</tr>
<tr>
<td>/3/</td>
<td>Description of the functions and operation of WinAC RTX V4.0. Available on the WinAC RTX V4.0 CD or directly from the control panel: Menu Help ‟Help on Controller.</td>
<td>SIMATIC WinAC RTX V4.0</td>
</tr>
<tr>
<td>/4/</td>
<td>Description or information on: General information on the PC tools Functions of NCM PC A&amp;D Service &amp; Support</td>
<td>Commissioning of SIMATIC NET PC Stations – Instruction and Quick Start for SIMATIC NCM PC / STEP 7 from version V5.2 and higher. Entry ID: <a href="#">13542666</a></td>
</tr>
<tr>
<td>/5/</td>
<td>Manual for industrial communication on PG/PC with SIMATIC NET. A&amp;D Service &amp; Support</td>
<td>SIMATIC NET – Industrial Communication with PG/PC Entry ID: <a href="#">16923753</a></td>
</tr>
<tr>
<td>/6/</td>
<td>Complete overview of the organization blocks (OB), system functions (SFC), system and standard function blocks (SFB) and standard function blocks as well as IEC functions contained in the operating systems of the CPUs of S7-300 and S7-400. A&amp;D Service &amp; Support</td>
<td>System Software for S7-300/400 System and Standard Functions Entry ID: <a href="#">1214574</a></td>
</tr>
</tbody>
</table>
### Bibliographic References

**Database Link to WinAC via OPC-XML**

| Entry ID: 21576581 |

| /7/ | Manual for SIMATIC Rack PC IL40S |
| A&D Service & Support |
| SIMATIC Rack PC IL40S Manual |
| Entry ID: 15317654 |

| /8/ | Installation instructions for CP5613 |
| A&D Service & Support |
| SIMATIC NET Product Information / Installation Instructions for CP5613, CP5614, CP5613 FO, CP5614 FO ... |
| Entry ID: 13664901 |

### Note

If the entries are not displayed immediately after clicking the links listed in the table above, click “Update” in your browser.