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 provisions do not imply a change in the burden of proof to your detriment.

The application examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. They do not represent customer-specific solutions. They are only intended to provide support for typical applications. Every user is responsible for proper operation of the described products. 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. It is not permissible to transfer or copy these application examples or excerpts of them without first having prior authorization from Siemens A&D in writing.

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:

- Basics of the software PLC SIMATIC WinAC RTX
- Basics of the open programming interface WinAC ODK
- Windows programming: Mechanisms for synchronization
- Windows programming: Mechanisms for data exchange

Delimitation

This application does not include a detailed description of

- SIMATIC Vision Sensors
- image processing mechanisms
- basics of Windows programming

Basic knowledge of these topics is required.

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 “SIMATIC Vision Sensor and WinAC ODK”, is displayed with a red margin.
Figure 1-1

Examples on PC-Based Automation

<table>
<thead>
<tr>
<th>Controlling</th>
<th>Visualizing</th>
<th>Technology</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automating with WinAC RTX</td>
<td>Interaction of WinCC and WinAC</td>
<td>Positioning with WinAC RTX</td>
<td>S7 communication with WinAC RTX</td>
</tr>
<tr>
<td>Installing the PC platform</td>
<td>Tuning for WinAC RTX</td>
<td>Tuning with Easy Motion Control</td>
<td>S7 communication via BSEND/BRCV</td>
</tr>
<tr>
<td>Process interface for WinAC RTX</td>
<td>Connecting WinCC to WinAC RTX</td>
<td>Connecting a MICROMASTER 440</td>
<td>via Industrial Ethernet</td>
</tr>
<tr>
<td>Configuration of WinAC</td>
<td>Visualizing with WinCC</td>
<td>Communication: WinAC ↔ S7-300</td>
<td></td>
</tr>
<tr>
<td>Getting Started</td>
<td>Getting Started</td>
<td>Getting Started</td>
<td>Getting Started</td>
</tr>
</tbody>
</table>

Examples from the classic world
- Data exchange via OPC XML
- Dynamic recipe data control
- Data exchange to the database via OPC XML
- ADO.NET database interface
- Application

Examples from the open world
- Data exchange via OPC XML
- Dynamic recipe data control
- Data exchange to the database via OPC XML
- ADO.NET database interface
- Application

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 informs you on the individual components and their interaction.
Structure of the document

This documentation 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></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>You can skip this part if you want to test the application first using the step-by-step instructions.</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” provides further information, e.g. bibliographic references.</td>
<td></td>
</tr>
</tbody>
</table>

Teaching material of this application

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

- Basics of PC-based control engineering
- Performance characteristics of SIMATIC WinAC RTX, the PC-based controller by Siemens
- Real-time capability with Windows operating systems
- Basics of the Windows operating system and its mechanisms for data exchange and synchronization
- Basics of the WinAC Open Development Kit (ODK), a programming interface to WinAC RTX

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.

Table of Contents

1 Automation Task ............................................................................................... 10
2 Automation Solution .......................................................................................... 11
  2.1 Overview of the overall solution ................................................................. 11
  2.2 Description of the functionalities of the application ................................. 13
  2.3 Alternative applications ........................................................................... 15
  2.4 Required components .............................................................................. 17
3 Performance Data ............................................................................................ 20
4 Function Mechanisms ..................................................................................... 22
  4.1 Basics: PC-based control with SIMATIC .................................................... 22
    4.1.1 SIMATIC WinAC basics ........................................................................ 23
    4.1.2 WinAC RTX basics .............................................................................. 23
    4.1.3 WinAC ODK basics ............................................................................ 26
    4.1.4 Principle of operation of Custom Code Extension (CCX) ....................... 28
  4.2 Basics: Software applications in Windows ................................................ 32
    4.2.1 Windows mechanisms for synchronization ............................................ 33
    4.2.2 Windows mechanisms for data exchange ............................................. 35
5 Installation of Hardware and Software ........................................................... 38
  5.1 Preparatory installation .............................................................................. 38
  5.2 Hardware installation ................................................................................ 39
  5.3 Installation of WinAC RTX ......................................................................... 40
  5.4 Installation of SPECTATION ..................................................................... 40
  5.5 Installation of the application software components .................................. 41
  5.6 Configuring the PC .................................................................................. 42
  5.7 Configuring and programming the vision sensor ....................................... 44
  5.8 Configuration and programming of WinAC RTX ...................................... 46
6 Operating the Application ............................................................................... 49
7 Explanation of the Codes of the Individual Elements ..................................... 56
  7.1 Initializing WinAC ODK CCX .................................................................... 56
  7.2 Example for a synchronous call of WinAC ODK CCX .............................. 58
  7.3 Example for an asynchronous call of WinAC ODK CCX ............................ 63
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4</td>
<td>Example for an ODK function which runs in the background</td>
<td>69</td>
</tr>
<tr>
<td>8</td>
<td>Adapting the Programs</td>
<td>74</td>
</tr>
<tr>
<td>8.1</td>
<td>Adapting the S7 program</td>
<td>74</td>
</tr>
<tr>
<td>8.2</td>
<td>Adapting the C++ applications</td>
<td>75</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Adaptations in the Windows application</td>
<td>75</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Adaptations in the ODK application</td>
<td>76</td>
</tr>
<tr>
<td>9</td>
<td>Bibliographic References</td>
<td>78</td>
</tr>
</tbody>
</table>
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 used components (standard hardware and software components and the additionally developed software).

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

Objective of Part A1

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

- Explanation of the automation problem
- 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>11</td>
</tr>
<tr>
<td>2.1</td>
<td>Overview of the overall solution</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Description of the functionalities of the application</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Alternative applications</td>
<td>15</td>
</tr>
<tr>
<td>2.4</td>
<td>Required components</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>Performance Data</td>
<td>20</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 hardware is to be used which has to be connected to the PC and which is to exchange data with the controller.

General task

Functionalities from existing, special Windows control programs (which are available as source code) are to be integrated into a SIMATIC control program. A suitable controller in this case is SIMATIC WinAC RTX.

General technological task

This application shows how SIMATIC WinAC RTX can interact with other, independent Windows applications. Windows standard mechanisms are to be used.

In general, this application focuses on linking a Windows application to the WinAC RTX controller using the WinAC Open Development Kit (ODK).

Specific technological task

The technological task is to monitor bottles on a conveyor belt with regard to their quality. A VS 723 Vision Sensor is to be used. It is to automatically perform the inspection and to provide good or bad information.

A Windows application (referred to as camera application in the following) is to run on the PC which visualizes the images of the vision sensor, controls it, and reports the results to WinAC RTX.

Solution requirements

- The vision sensor is to be fully operable from the camera application.
- All functions also have to be operable from the S7 program in WinAC RTX.
- Two different bottle types have to be configurable.
- The result of an inspection has to be reported to WinAC RTX as binary value.
- Images of faulty test objects are to be stored on hard disk.
- Standardized communication mechanisms are to be used.
- Inspection tasks are to be processed independently by the vision sensor.
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 only need 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>2.1</td>
<td>Overview of the overall solution</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Description of the functionalities of the application</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Alternative solutions</td>
<td>15</td>
</tr>
<tr>
<td>2.4</td>
<td>Required components</td>
<td>17</td>
</tr>
</tbody>
</table>

2.1 Overview of the overall solution

Display of the components involved

The following overview shows the hardware configuration of the example application and the respective standard and user software components. Please note that you can start up the application also without camera. A simulation mode is available for this purpose.
Note: You do not necessarily require the VS 723 Vision Sensor to start this example. A simulation mode is available in the Windows application.
2.2 Description of the functionalities of the application

Solution elements of the application

The application consists of several solution elements which are closely linked.

Figure 2-2

<table>
<thead>
<tr>
<th>No.</th>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SIMATIC WinAC RTX</td>
<td>A controller with real-time capability. It takes the actual control of the system.</td>
</tr>
<tr>
<td>2</td>
<td>Camera</td>
<td>Inspects the quality of glass bottles and reports the test result to the PC via a TCP/IP connection.</td>
</tr>
<tr>
<td>3</td>
<td>Camera application</td>
<td>The camera application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• controls the camera (with a software component called ActiveX Control) and reads the test results of the camera</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• exchanges data with WinAC RTX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• stores images of the camera on the hard disk.</td>
</tr>
<tr>
<td>4</td>
<td>WinAC ODK application</td>
<td>The ODK application transfers the data between the camera application and WinAC RTX and vice versa. In this example, it does not contain a control functionality. However, this would be possible.</td>
</tr>
</tbody>
</table>
Sequence of selected core functionalities

The tables below describe which steps are executed in the application to ensure that specific functionalities are processed:

Setting the status of the camera

Table 2-2

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WinAC RTX sends a command to the ODK application, which contains data on the mode to be set.</td>
<td>The command is transmitted via an SFB call in WinAC RTX and the data are transferred in an ANY pointer.</td>
</tr>
<tr>
<td>2</td>
<td>The ODK application receives the job and buffers the data. A simultaneously running program part is started.</td>
<td>Program parts which can run simultaneously are referred to as “threads” or “processes” in Windows terminology.</td>
</tr>
<tr>
<td>3</td>
<td>The simultaneously running program part sends a Windows message with data to the camera application.</td>
<td>Windows messages are sent to a specified receiver and delivered by the operating system.</td>
</tr>
<tr>
<td>4</td>
<td>The camera application receives the Windows message and stores the data.</td>
<td>Requires that a message handler for Windows messages is defined in the application.</td>
</tr>
<tr>
<td>5</td>
<td>Via ActiveX Control, the camera application sends the command to the camera.</td>
<td>ActiveX Control is a software component which was specially developed for the communication with SIMATIC Vision Sensors.</td>
</tr>
</tbody>
</table>

Reading the number of inspected bottles from the camera

Table 2-3

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>After each inspection, the camera sends the number of inspected bottles to the camera application</td>
<td>This communication is again performed via ActiveX Control.</td>
</tr>
<tr>
<td>2</td>
<td>The camera application writes this counter value into the shared memory</td>
<td>In Windows the shared memory is handled like a file.</td>
</tr>
<tr>
<td>3</td>
<td>The ODK application cyclically checks whether the value of the bottle counter in the shared memory has increased. The value is read.</td>
<td></td>
</tr>
</tbody>
</table>
Overview

SIMATIC Vision Sensors and WinAC ODK

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>The read value is directly written into a memory variable in WinAC RTX by the ODK application.</td>
<td>The ODK application has access to almost all memory areas in WinAC RTX.</td>
</tr>
</tbody>
</table>

Advantages of the solution example

- Any hardware and software components can be integrated into SIMATIC control programs
- The camera application can very easily be replaced by any other Windows application
- Via the shared memory, high-performance exchange of large data amounts is possible
- Use of all performance characteristics offered by the Windows operating system
- The solution includes an easy-to-use visualization
- Images of faulty components are stored on the hard disk for later error analysis.

Advantages of PC-based automation by Siemens

- Cost saving, since the controller runs on the PC and no external controller is required
- Use of existing, standardized hardware and standardized interfaces
- High flexibility and low costs
- Easy interfacing of the controller to standard software (e.g. Microsoft Office) and own applications (VB, C++, C#, ...)
- Use of advancing innovations in the PC sector
- Use of known engineering tools (e.g. STEP 7)
- Optimal integration of controlling, visualization, communication, and technology

2.3 Alternative applications

Alternative software solutions

With the displayed mechanisms, any software module can be integrated into the control cycle of WinAC RTX, e.g.:

- Controllers (e.g. PI controllers, implemented in C++)
- Interfacing to databases
- Measurement data acquisition software
Statistics tools

Alternative hardware solutions

Hardware components by Siemens and third-party manufacturers can communicate with WinAC RTX, e.g.:

- Scanners
- Measurement data acquisition cards
- Modems
- Robots
2.4 Required components

The hardware and software components for the respective stations are listed in the tables below. 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 this application, you only require an industrial PC and optionally a camera of the Machine Vision series.

Table 2-4

<table>
<thead>
<tr>
<th>Component</th>
<th>No.</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>6AG4011-0CA21-0JX0</td>
<td>Configurator: See FAQ ID 17128155</td>
</tr>
<tr>
<td>SIMATIC VS723 Vision Sensor</td>
<td>1</td>
<td>6GF1723-0AA</td>
<td>Optional; 640 x 480 pixels b/w</td>
</tr>
<tr>
<td>Power cable for VS72x Vision Sensor</td>
<td>1</td>
<td>6GF9002-2AD</td>
<td>Only if VS723 is used</td>
</tr>
<tr>
<td>Ethernet cable cross-over</td>
<td>1</td>
<td>Depending on manufacturer</td>
<td>Only if VS723 is used</td>
</tr>
<tr>
<td>Lens XENOPLAN 1.4/23 MM for VS72x</td>
<td>1</td>
<td>6GF9001-1AL</td>
<td>Only if VS723 is used</td>
</tr>
<tr>
<td>PS307 2A power supply</td>
<td>1</td>
<td>307-1BA00-0AA0</td>
<td>Only if VS723 is used; only 250 mA are required.</td>
</tr>
<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

Table 2-5

<table>
<thead>
<tr>
<th>Component</th>
<th>No.</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></td>
</tr>
<tr>
<td>SIMATIC NET IE SOFTNET-S7 V6.2</td>
<td>1</td>
<td>6GK1716-0HB62-3AA4</td>
<td>WinAC RTX cannot be installed without SIMATIC NET</td>
</tr>
<tr>
<td>STEP 7 V5.3</td>
<td>1</td>
<td>6ES7810-4CC07-0YA5</td>
<td>Includes NCM S7 Ethernet</td>
</tr>
<tr>
<td>SPECTATION 2.6 configuration software</td>
<td>1</td>
<td>6GF8007-3AA26</td>
<td>Only if VS723 is used</td>
</tr>
</tbody>
</table>
SIMATIC Vision Sensors and WinAC ODK

Application software components

This example application consists of the following components.

For further information on the installation of hardware and software please refer to the chapter “Installation of hardware and software”.

Table 2-6

<table>
<thead>
<tr>
<th>Component</th>
<th>Hinweis</th>
</tr>
</thead>
<tbody>
<tr>
<td>21572937_WinAC_TO_CODE_v11.zip</td>
<td>This file contains a setup. It installs the components listed in the following.</td>
</tr>
<tr>
<td>ODK_Application</td>
<td>The source code for the creation of the ODK program created with the ODK wizard and implemented in C++; the completely generated library is also included.</td>
</tr>
<tr>
<td>STEP7_Prj</td>
<td>S7 project with configuration and code, which can be directly loaded to WinAC RTX.</td>
</tr>
<tr>
<td>Cam_Config</td>
<td>The configuration of the VS723 Vision Sensor; can be loaded directly to the sensor with SIMATIC Spectation.</td>
</tr>
<tr>
<td>Cam_Application</td>
<td>The source code and the executable file of the camera application.</td>
</tr>
<tr>
<td>VS7x_ActiveX</td>
<td>An ActiveX Control used to control the camera. It can be integrated into Visual Basic or C++ projects very easily.</td>
</tr>
</tbody>
</table>

| 21572937_WinAC_TO_v11_DOKU_e.pdf | This document |

Effort required for programming/configuring

- Assuming average SIMATIC knowledge, one man-day is required for the programming of the S7 program to implement a comparable application.
- With good Windows programming knowledge, approx. 12 man-days are required to program the camera application.
- Assuming good Windows programming knowledge, the creation of the ODK source code takes approx. 5 man-days.
- With good basic knowledge, the configuration of the SIMATIC VS 723 Vision Sensor requires one man-day.
- Approx. 4 days are to be estimated for the integration of all components.
The periods estimated above are rough guide values and may be considerably larger or smaller depending on the previous knowledge. When performing such projects it is required that S7 programmers work together closely with the Windows programmers, since both are strongly linked.
### 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>Max. data size during transfer from WinAC to ODK application</td>
<td>64 kB</td>
<td>Max. size of a DB in WinAC RTX</td>
</tr>
<tr>
<td>Max. data size during transfer from ODK application to Windows application and vice versa</td>
<td>2 GB</td>
<td>When using the shared memory; max. size of a file in Windows 32</td>
</tr>
<tr>
<td>WinAC RTX response time to interrupts from the ODK application</td>
<td>In the μs range</td>
<td>Depending on the processor speed and the running software</td>
</tr>
<tr>
<td>Response time of Windows programs to messages from other programs</td>
<td>10 –100 μs</td>
<td>Measured with the SendMessage¹ function; depending on the processor speed and the running software</td>
</tr>
</tbody>
</table>

¹ SendMessage is a Windows function which sends messages to other Windows programs and which blocks the sender until the receiver has received the message. PostMessage has the same function, but it does not block the sender.

However, it is not advisable to use the PostMessage function since the runtime of a message very strongly depends on the number of currently opened Windows applications and on the system utilization.
Part A2: Function Mechanisms

Objective of Part A2

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

- Explanation of all function elements involved
- Illustration of the reusable components
- Respective background knowledge.

Contents of Part A2

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Function Mechanisms</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Basics: PC-based control with SIMATIC</td>
<td>22</td>
</tr>
<tr>
<td>4.2</td>
<td>Basics: Software applications in Windows</td>
<td>32</td>
</tr>
</tbody>
</table>
4 Function Mechanisms

What will you find here?

You will find information on

- the basics of PC-based control engineering
- the WinAC RTX controller
- the WinAC Open Development Kit (WinAC ODK)
- relevant Windows mechanisms (synchronization and data exchange)
- the interaction of the individual components.

What can you do with it?

Basically, this application can be started immediately. The installation instructions tell you how to start it 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.

Structure of this chapter

For orientation, the structure of this chapter is shown in the table below.

Table 4-1

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Function Mechanisms</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Basics: PC-based control with SIMATIC</td>
<td>22</td>
</tr>
<tr>
<td>4.2</td>
<td>Basics: Software applications in Windows</td>
<td>32</td>
</tr>
</tbody>
</table>

4.1 Basics: PC-based control with SIMATIC

Overview

This chapter describes the mechanisms on which the example application is based. These mechanisms are displayed in a generalized form to ensure that they can be easily transferred to similar applications.

You learn what WinAC ODK is, how the WinAC RTX controller exchanges data with the ODK program and how this program exchanges data with a Windows application via Windows standard mechanisms.
SIMATIC Vision Sensors and WinAC ODK

4.1.1 SIMATIC WinAC basics

Products of the SIMATIC WinAC series
The SIMATIC WinAC series consists of the following products:
- WinAC RTX
- WinAC Slot
- WinAC ODK

Of course all these products are fully integrated into the Totally Integrated Automation concept (TIA) and thus programmable with STEP 7.

Since the products are pure software products, the term WinAC software comprises WinAC Basis, WinAC RTX and WinAC PN.

Description of WinAC RTX
In addition to the advantages of WinAC Basis, the WinAC RTX software controller offers deterministic properties. This is ensured by Windows RTX Realtime Extension.

This application example is based on WinAC RTX. The following chapters provide detailed information on this product.

WinAC Slot
In the two versions 412 and 416, WinAC Slot is a PC-based controller in form of a PCI card. This card already features a standard integrated DP and MPI interface. The connection to the optional Power Supply Extension Board provides the controller with additional protection against a failure of PC hardware and software.

WinAC ODK
WinAC Open Development Kit is a software product which offers interfaces between Windows applications and WinAC controllers. This enables to create Windows applications exchanging data in both directions with WinAC controllers in a way that is as easy as possible.

This application focuses on WinAC ODK.

4.1.2 WinAC RTX basics

What is WinAC RTX?
The Windows Automation Center with real-time extension (WinAC RTX) offers the functionality of a programmable logic controller (PLC) in a PC-based environment, e.g. Windows XP. With regard to its functions, WinAC RTX can be compared to an S7-400.

WinAC RTX controller
WinAC RTX executes the S7 program on the PC and controls the connected distributed I/O. WinAC RTX is fully code-compatible with other
SIMATIC S7 controllers. This ensures that existing S7 programs of an S7-300 or S7-400 controller are also executable on WinAC RTX.

**RTX real-time extension**

WinAC RTX uses the real-time extensions (RTX) for Windows by Ardence (formerly VenturCom). These extensions enable the following functions:

- Deterministic operation with predictable response times.
- Isochronous mode (equidistance, constant bus cycle time)
- In case of a Windows system failure ("blue screen"), the process control of WinAC RTX keeps running independent of the operating system and can thus be properly shut down (via interrupt OBs).

**Software architecture**

The figure below shows the configuration of a WinAC RTX system. It illustrates the separation between the Windows operating system and the RTX real-time system by Ardence on which WinAC RTX is based.

**Process interface for WinAC RTX**

PROFIBUS DP forms the process interface of WinAC RTX. PROFIBUS communications processors (e.g. PCI card CP 5613, in the future also CP 5611) are plugged into the PC and configured as interface modules (IF modules) in HW Config (same as for S7-400).
The CPs configured as IF modules are addressed via real-time hardware drivers. This enables the isochronous mode (equidistance of the bus cycles) for the connected PROFIBUS-DP master systems.

Up to four CPs can be configured as IF modules for WinAC RTX.

System communication with WinAC RTX

The system communication with WinAC RTX is performed via CPs which are plugged into the PC. These CPs are configured via HW Config of the SIMATIC PC station.

S7 communication is used for the system communication of SIMATIC S7 stations. To use S7 communication, it is required to install and to license the SIMATIC NET PC software on the PC on which WinAC RTX runs (the delivery of WinAC RTX includes the latest SIMATIC NET CD).

PG routing with WinAC RTX

WinAC RTX supports PG routing. This ensures that STEP 7 can reach S7 stations in other subnets via the CPs of a WinAC PC station.

The WinAC RTX control panel

The user interface of the WinAC RTX controller resembles the appearance of a classic S7-400 controller. It features the same status LEDs as well as buttons to change the mode.

The control panel can be shut down during operating WinAC RTX, the controller keeps running. To restart the control panel, double-click in the task bar.

Figure 4-2
4.1.3 WinAC ODK basics

Functionality of WinAC ODK

The WinAC Open Development Kit is a software tool which enables the creation of a Windows application (in the following referred to as ODK application); the functions of this application can be called directly from WinAC RTX. This enables the integration of the complete performance range offered by the Windows operating system and applications based on it into a WinAC control program.

ODK generates a basic framework in the C++ programming language in which the user can program his desired functionalities. This basic framework considerably facilitates the lead-in to the programming. However, it does not necessarily have to be used.

What is WinAC ODK V 4.1?

WinAC ODK V 4.1 is the combination of the two previous products WinAC Slot T-Kit and WinAC ODK V 4.1. Now you only require one more product for the programming of Windows applications for WinAC Slot and WinAC software (Basis and RTX).

Components of WinAC ODK V 4.1

WinAC ODK V 4.1 includes three components which will be described in the following:

- Custom Code Extension Interface (CCX)
- Shared Memory Extension Interface (SMX)
- Controller Management Interface (CMI)

Custom Code Extension Interface (CCX)

CCX is the successor of WinAC ODK V 4.1. It enables the access to data in WinAC RTX from the S7 program to the ODK application and in opposite direction. The S7 programmer uses SFB calls for these accesses.

You use CCX if you want to call C++ functions directly and as part of the control cycle of WinAC RTX. This application is based on CCX which will be described in detail. It can only be used together with WinAC RTX.

Shared Memory Extension Interface (SMX)

SMX replaces WinAC T-Kit. Interfaces are provided, which enable the data exchange between WinAC and the ODK application. To the S7 programmer, this data exchange appears as access to I/O data.

You use SMX particularly for PC applications performing high-performance exchange of large data amounts with WinAC. It can be used in WinAC RTX as well as in WinAC Slot.
Figure: Components of WinAC ODK V 4.1

The following figure shows which components can be used with which WinAC version.

Figure 4-3

Delivery scope of WinAC ODK V4.1

WinAC ODK consists of the following parts:

- A wizard which, based on the input of the user, automatically generates a basic framework in the C++ programming language. This takes place optionally for CCX or SMX.
- An S7 library with two S7 blocks (SFB 65001 and 65002) which enable the initialization and the call of the ODK applications. They are executable directly in WinAC RTX and only required for CCX.
- Header files which are required for the programming in C++.
- Libraries and header files for CMI.
- A detailed documentation in electronic form and example programs.
4.1.4 Principle of operation of Custom Code Extension (CCX)

Graphic display of the principle of operation of CCX

The figure shows how the ODK application (which includes different ODK functions) is connected with WinAC and other Windows applications. It also shows how external hardware can be used using ODK.

The violet Windows application is optional. The developer has the option to implement his functionality directly in the ODK application, which is e.g. useful for controllers or database applications. The Windows application is then no longer required.

However, it is frequently necessary to include existing Windows applications in the control task. The solution with an individual Windows application is recommended for programming reasons also for applications which feature visible user interfaces (windows).

Figure 4-4

The ODK application from the WinAC project planner’s point of view

The calls of ODK functions can be compared to calls of SFBs and OBs in a classic controller. There are three different call types:

- Synchronous call of an ODK function: The function is already completely processed after one call.
- Asynchronous call of an ODK function: The function requires several calls until it is completely processed.
- Execution of continuous ODK functions in the background (monitoring). The function runs permanently in the background and executes monitoring functions.
Only the type of the implementation in the ODK application decides which of the three function types is available.

A special case of the call is the initialization of the ODK application. It is performed once before using ODK functions for the first time.

Initialization and the three call modes will be described in the following sections.

Initializing the ODK application

By calling SFB 65001 specially provided for this purpose, the ODK application is created and initialized. After successful initialization, this SFB returns a unique ID (status) which has to be stored for all following calls of ODK functions.

Initializing the ODK application: Screen shot

The screen shot below shows the call for initializing an ODK application using SFB 65001.

General information on calling ODK functions

For WinAC, the call of a function in the ODK application is identical to the call of an SFB:

1. SFB 65002 is e.g. supplied with input parameters and called in OB1
2. SFB 65002 initiates the call of a function in the ODK application (ODK function)
3. The ODK function is running. During this time the execution of OB1 is interrupted
4. After completing the ODK function, the respective output parameters are set and the control goes back to SFB 65002
5. SFB 65002 receives the return parameters from the ODK function and transfers them to OB 1. OB 1 is continued
Calling ODK functions: Screen shot

The screen shot shows the call of SFB 65002 with the function 1 (Command = DW#16#1) in the ODK application. Any number of functions can be implemented.

Figure 4-6

Network 2: Get status

Receives the status from the ODK application

The synchronous ODK call

The following time strip shows the synchronous call of an ODK function. OB1 calls SFB 65002 (with IN parameters) which executes the ODK function. The processing of OB1 is interrupted as long as this function is active. At the end, the ODK function returns its result to OB1 as OUT parameter.

The synchronous call is only suitable for ODK functions which are short with regard to time, since otherwise a cycle time exceeding in OB1 may occur.

Figure 4-7
The asynchronous ODK call

The following time strip shows the asynchronous call of an ODK function. OB1 calls SFB 65002 (with IN parameters) which starts the ODK function. The SFB call returns immediately. The processing of OB1 is only interrupted for a very short period of time. The ODK function runs “quasi simultaneously” to OB1 in the background.

OB1 cyclically polls the status of the ODK function. As long as the function reports BUSY a result is not yet available.

When the ODK function is finished, it reports its result back to OB1 as OUT parameter in the following call from OB1.

The asynchronous call is also suitable for ODK functions which are long with regard to time, since only an insignificant extension of OB1 cycles occurs.

Figure 4-8

Executing ODK functions in the background

The time strip below shows the continuous function of ODK. This function runs permanently in the background of the ODK application. It thus runs “quasi simultaneously” to all program parts in WinAC RTX. The continuous function is started during initializing the ODK program, it is stopped when the controller is set to STOP mode.

The continuous function is typically used for monitoring functions in the background. It is similar to OB 90 in a classic controller. A typical application would e.g. be the polling of the serial interface in the PC to determine whether new data are available.

Figure 4-9
Data exchange with ODK in continuous functions

To ensure that the continuous function can deliver data to WinAC RTX, it can request an asynchronous OB in WinAC RTX (e.g. OB52 provided specially provided for this purpose, see figure above) and transfer data to it. The response of WinAC RTX to such a call is freely programmable.

As a further alternative, the ODK function can e.g. directly set the variables in WinAC RTX, e.g. memory words. The response also has to be programmed in the S7 program.

To transfer data from WinAC RTX to a continuous function, a synchronous call as described above can be used.

4.2 Basics: Software applications in Windows

Stand-alone Windows applications

Stand-alone Windows applications (referred to as Windows application in the following) are Windows programs which are executable independently as a closed unit on a Windows operating system. These programs can either be purchased as product of a manufacturer (e.g. Adobe Acrobat Reader) or developed individually (e.g. own Windows programs containing control algorithms for the control of processes).

The delivery of this application includes a Windows application which communicates with a SIMATIC VS 723 Vision Sensor and which visualizes its images (see Figure 4-4). It can also simulate the vision sensor.

Windows processes

A Windows application which is currently executed by the operating system is referred to as process in the Windows programming language. A remarkable feature of a process is the fact that it has an individual, protected memory area, a defined priority and that it is cyclically processed by Windows (more precisely: By the scheduler).

Windows threads

A thread is the part of a process which can be executed independent of all other parts of the process. For example it is possible to wait for data from the serial interface in a thread.

A thread is the smallest executable unit which the operating system can process "quasi simultaneously".

Data exchange in Windows

To perform their tasks, it is required that threads, processes and also different computers exchange data with each other and also with the operating system. Data exchange between processes is referred to as interprocess communication. The data exchange with the operating system is performed via operating system calls. Windows makes available a number of mechanisms which will be described in the following.
PC-internal and external hardware can also be accessed via Windows mechanisms.

If time dependencies exist, different processes must synchronize. A number of Windows functions is available for this synchronization.

Windows resources
A term frequently used in programming in Windows is “resource”. Resources in Windows can be:

- Hardware components
- Memory areas (also files)
- Interfaces (e.g. serial interface)
- Virtual devices (e.g. the console)

4.2.1 Windows mechanisms for synchronization

Overview
The Windows operating system offers the following mechanisms for synchronization between processes:

- Mutexes
- Semaphores
- Events

Mutexes
A mutex (mutual exclusion) is an object which secures exclusive access to a commonly usable resource. A mutex can be locked and released. This is necessary, since processes and threads run simultaneously in Windows. It may e.g. occur that two simultaneous processes simultaneously request one resource. This can be prevented by the use of mutexes.

Figure: Principle of operation of mutexes
The graphic below shows how two processes use a shared memory for data exchange. To secure data consistency, it has to be ensured that always only one process has access to the memory at a given time. This can be realized by locking mutexes. If a mutex is already locked by process 1 and if process 2 also tries to lock it, process 2 waits until the mutex is released by process 1. This mechanism is made available by Windows.
Semaphores

Semaphores function similarly to mutexes, but they feature an integrated counter. This enables to define the number of processes which can simultaneously access a resource. If the semaphore is e.g. initialized with the value 2, this means that two processes can use this resource simultaneously. If a third process requests the semaphore, this process must wait until one of the two other processes releases the semaphore.

Principle of operation of semaphores: Figure

The graphic below shows a buffer which includes two data records (DR1, 2). Each data record can be assigned by one process. Thus two processes can access the buffer simultaneously.

This problem is solved using a semaphore which is initialized with the value 2. Depending on the locking of the semaphore, the value is decremented by one until the counter reaches zero. Processes which then try to lock the semaphore have to wait until another process releases the semaphore.

Events

An event is a Windows event which knows two statuses: Signaled and not signaled. Events are used for notification between different processes.
**Principle of operation of events: Figure**

The following graphic shows how process 1 writes data into a buffer and informs process 2 that the data are ready. This notification is performed via an event. It is required that process 2 waits for an event. Process 2 can reset the event either immediately or after completing the read operation.

Figure 4-12

**4.2.2 Windows mechanisms for data exchange**

**Overview**

The Windows operating system offers the following mechanisms for data exchange between processes:

- Windows messages
- Files (on the hard disk as well as in the memory)
- Via communication channels

**Windows messages**

Windows messages are suitable for the transfer of small data amounts (several bytes). Windows makes available two functions:

- **SendMessage**: Sends a message to a process and waits until the message has been received.
- **PostMessage**: Like SendMessage but does not wait.

Both functions send a message to a process which can be selected (more precisely: To its window). This process must have a function which accepts and edits this message (a "message handler").
Files

Data amounts of any size can be transferred via files. It makes no difference to Windows whether the file is on the hard disk or in the main memory.

Data exchange via files: Figure

Data exchange between files works between any number of processes. Each process which wants to use the file has to open it first and can then read or write it depending on the authorization. To ensure data consistency, it is required to use the synchronization mechanisms described above.

Figure 4-14
Part B: Installation of the Example Application

Overview

Contents of Part B

Part B takes you step by step through configuration and startup of the application.

Objective of Part B

Part B of this document provides the reader the following:

- Explanation of the installation of the example with all hardware and software components
- Information on operating 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>40</td>
</tr>
<tr>
<td>5.4</td>
<td>Installation of SPECTATION</td>
<td>40</td>
</tr>
<tr>
<td>5.5</td>
<td>Installation of the application software components</td>
<td>41</td>
</tr>
<tr>
<td>5.6</td>
<td>Configuring the PC</td>
<td>42</td>
</tr>
<tr>
<td>5.7</td>
<td>Configuring and programming the vision sensor</td>
<td>44</td>
</tr>
<tr>
<td>5.8</td>
<td>Configuring and programming WinAC RTX</td>
<td>46</td>
</tr>
<tr>
<td>6</td>
<td>Operating the Application</td>
<td>49</td>
</tr>
</tbody>
</table>
5 Installation of Hardware and Software

Introduction
This chapter describes the installation of the components for this application. It is divided into the following sections.

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>40</td>
</tr>
<tr>
<td>5.4</td>
<td>Installation of SPECTATION</td>
<td>40</td>
</tr>
<tr>
<td>5.5</td>
<td>Installation of the application software components</td>
<td>41</td>
</tr>
<tr>
<td>5.6</td>
<td>Configuring the PC</td>
<td>42</td>
</tr>
<tr>
<td>5.7</td>
<td>Configuring and programming the vision sensor</td>
<td>44</td>
</tr>
<tr>
<td>5.8</td>
<td>Configuring and programming WinAC RTX</td>
<td>46</td>
</tr>
</tbody>
</table>

Installation sequence
A specific installation sequence should be followed to ensure correct installation of the SIMATIC components. The components according to their installation sequence are listed in the following table.

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STEP 7 V5.3 (or higher), see 5.1 Preparatory installation</td>
</tr>
<tr>
<td>2</td>
<td>SIMATIC NET PC software V6.2 (or higher), see 5.1 Preparatory installation</td>
</tr>
<tr>
<td>3</td>
<td>Installation of the VS 723 Vision Sensor, see 5.2 Hardware installation</td>
</tr>
</tbody>
</table>
| 4   | WinAC RTX V4.1 (or higher), see 5.3 Installation of WinAC RTX  
   This installation is divided into the following:  
   1. Installing and checking the Ardence RTX extensions  
   2. Installing and authorizing the WinAC RTX software  
   3. Installing required service packs for WinAC RTX |

5.1 Preparatory installation

Introduction
It is absolutely necessary to install STEP 7 and the SIMATIC NET software before installing WinAC RTX.
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 RTX is to run. At this point, the installation of STEP7 will not be described. The installation is self-explanatory and takes place in the usual Windows environment.

SIMATIC NET

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

From STEP 5.2 version 7 on, “Advanced PC Configuration” is used to put PC stations into operation. It enables the configuration of PC stations directly in STEP 7. Prior to the use of “Advanced PC Configuration”, we strongly recommend reading manual /2/ “SIMATIC NET Commissioning PC Stations – Quick Start”.

5.2 Hardware installation

Establishing a data connection via TCP/IP
Connect the TCP/IP port of the camera to the TCP/IP port of your PC using the Ethernet cable. If this port is already reserved for a network connection, you can plug an additional network card into the PC.

Connecting the power supply of the camera
Electrically isolate the power supply from the line supply. Connect the power supply cable to the power supply according to the vision sensor manual. Connect the other end of the cable to the vision sensor.

Mounting the lens
Screw the lens onto the vision sensor.

Mounting the camera
Mount the camera vertically at a fixed object so that the lens is located approx. 22 cm above the upper edge of the table.
5.3 Installation of WinAC RTX

**Note**
Administrator rights ("ADMIN") for your operating system are required for the installation of WinAC RTX V4.1.

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

### Table 5-2

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The setup program starts automatically after inserting the WinAC RTX CD. If the program does not start automatically, execute the “Setup.exe” 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>
</tbody>
</table>
| 3   | Click the “install VenturCom RTX” button and follow the instructions in the dialog box. Ardence (formerly VenturCom) RTX is installed on your PC.  
**Note**  
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.  
**Note**  
If the error message “Your System is using a HAL that is not supported by RTX 5.12” is displayed during the installation of the Ardence RTX extensions, please read the FAQ with the ID [517053416](http://www.ad.siemens.com/support) on the A&D Support home page (www.ad.siemens.com/support). |
| 4   | After restarting the PC, the functionality of the Ardence RTX extensions has to be checked. Click the point “Step 2” in the “WinAC RTX V4.1 Setup” dialog. You receive instructions for testing the RTX extensions. |
| 5   | By clicking the point “Step 3” in the installation dialog, the installation of the WinAC RTX software is started. |

5.4 Installation of SPECTATION

**Note**  
SPECTATION only has to be installed if you want to connect the VS 723 Vision Sensor.

### Use of the Spectation software

Spectaion is used for programming and configuring vision sensors. In this application, the camera was programmed in such a way that it can detect and inspect two different bottle types. An image of a “good” bottle is shown to the camera and stored. To compensate tolerances and different lighting conditions, it is possible to configure the variable of the maximum permissible deviation.
The system data and programs are loaded to the vision sensor with Spectation.

Installation of the Spectation software
Spectation is a software which can run on the operating systems Windows ME, 2000 and XP. Install the software like other Windows software.

5.5 Installation of the application software components

Copying the source code and the data
You can download the individual components of the application including the source code from the same web site from which you downloaded this document. They can be installed via a setup.

First open the ZIP archive “21573047_WinAC_TO_CODE_v10_e.zip” and extract the contained files to a directory of your choice.

Among other things, the extracted files include the Setup.exe file. Start the file by double-clicking it.

A setup program opens. Follow the instructions of this program. If you are asked for a directory for the installation, please enter C:\WinAC_ODK_Sample. This prevents later changes in the STEP 7 program.

Installing ActiveX Control
The above setup calls a further setup program which installs ActiveX Control for VS 723. Follow the instructions of the program. If you are asked for a path, you can either use the suggested path or enter any path.

File structure after the installation
The file structure after the installation looks as follows:
Contents of the individual folders

The folders in C:\WinAC_ODK_Sample have the following contents:

<table>
<thead>
<tr>
<th>Folder</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cam_Application</td>
<td>Contains the C++ source code of the camera application as Visual Studio.net project.</td>
</tr>
<tr>
<td>Cam_Application\Release</td>
<td>Contains the executable camera application.</td>
</tr>
<tr>
<td>Cam_Application\res</td>
<td>Contains resource files which are a part of the source code.</td>
</tr>
<tr>
<td>Cam_Config</td>
<td>Contains the configuration and the program for the camera. In addition, you find a Word file with the image of two bottles intended for printout.</td>
</tr>
<tr>
<td>ODK_Applikation</td>
<td>Contains the C++ source code of the ODK application as Visual Studio.net project.</td>
</tr>
<tr>
<td>ODK_Applikation\Release</td>
<td>Contains the complete library of the ODK application which can be loaded from WinAC RTX. This path is stored as string in data block 652 of the STEP7 program.</td>
</tr>
<tr>
<td>STEP7_Prj</td>
<td>Contains the archived STEP 7 project</td>
</tr>
<tr>
<td>VS72x_ActiveX</td>
<td>Contains the setup program of ActiveX Control for VS 723. It has already been installed on your computer with the above setup.</td>
</tr>
</tbody>
</table>

5.6 Configuring the PC

Configuration of the network connection

To be able to address the vision sensor, it is first required to adapt the network connection of your PC. Please perform the following settings.
### Table 5-4

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Click the “Network Neighbourhood” icon on your desktop with the right mouse button and click “Properties”. A new window opens.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>In this window, select the LAN connection via which you connect yourself to the vision sensor. Click it with the right mouse button and click “Properties”. A new window opens.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Select “Internet Protocol” and click the “Properties” button. A new window opens. In this window, select the option “Use the following IP address”. Enter 192.168.0.220 as IP address. 255.255.255.0 is automatically entered as subnet mask.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Close all dialog boxes by clicking OK.</td>
<td></td>
</tr>
</tbody>
</table>
5.7 Configuring and programming the vision sensor

Establishing the connection to the vision sensor
To be able to load the program to the vision sensor, it is first required to establish a connection to the vision sensor.

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start SPECTATION via Start→SIMATIC→Machine Vision→Spectation</td>
<td>The Spectation main window opens.</td>
</tr>
<tr>
<td>2</td>
<td>Open the “Comm” menu and select “PC communication”. A window opens.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Select the “Network neighbourhood” folder. After searching for several seconds, the vision sensor is displayed.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Select the found vision sensor and click “Edit…”. The “IP configuration” window opens.</td>
<td></td>
</tr>
</tbody>
</table>
Part B: Installation of the Example Application

Overview

SIMATIC Vision Sensors and WinAC ODK

5 Enter a name for your vision sensor. Change the IP address to 192.168.0.224. Change the subnet mask to 255.255.255.0.

Note:
Vision sensor and Ethernet interface of the PC have to be located in the same subnet. In our example that means that the first three numerical pads of the IP address have to be identical and that the last numerical pad must be different.

6 Exit the dialog box by clicking OK. Select the vision sensor and click “Connect”. If everything has been done correctly, the “image display” window is displayed. Click the “Start” button and an image is displayed.

Loading the configuration to the vision sensor

To ensure that the vision sensor can perform its test tasks, it is required to parameterize and program it. Proceed as follows:

Table 5-6

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>You are still in the SPECTATION program. In the “testing scripts” menu, select “Backup System from PC”. A new dialog appears. Scroll to the “Cam_Config” folder which you have installed for this application. Open the “BottleCheck.dvt.prod720” file. If this file was created with an older version of Spectation, please confirm the corresponding window. Please answer the question whether you want to save all data in the flash memory by clicking “Yes”. The program is now directly transferred to the vision sensor.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>In the “testing scripts” menu, select “backup system data from PC”. A new dialog appears. Again scroll to the “Cam_Config” folder and open the “BottleCheck.sys720” file.</td>
<td></td>
</tr>
</tbody>
</table>
9 In the following window, do not modify the specified values and click “OK”. If the file was created with an older version of Spectation, please confirm the corresponding window. Please answer the question whether you want to save all data in the flash memory by clicking “Yes”. The system data are now directly transferred to the vision sensor.

10 Open the Word file “Flaschen.doc” in the “Cam_Config” directory and print it. It contains images of a wine and a water bottle with which the camera was parameterized. Fold the printout so that you can place it on the table.

11 For testing, you can place the image of a wine bottle approx. 40 cm in front of the camera. Focus the lens. Monitor the result of the soft sensor in the result table of SPECTATION. If you capture the image of the bottle in the correct distance and the correct angle of rotation, you receive the result “Pass”. The correct position of the camera has to be determined by trying. The vision sensor is now ready for use.

5.8 Configuration and programming of WinAC RTX

Important settings in the Station Configuration Editor

Open the Station Configuration Editor via:

Start → Station Configuration Editor

or by double-clicking in the task bar.

To apply the example project for this application, the following settings should be made in the Station Configuration Editor:
Part B: Installation of the Example Application

Overview

SIMATIC Vision Sensors and WinAC ODK

- Change the station name to “PCWinAC” using the “Station Name” button. This is the name assigned for the WinAC station in the STEP 7 project.
- WinAC RTX should be on slot (index) 2.

Retrieving the S7 program

Open STEP 7 and retrieve (File → Retrieve) the archive you have installed with this application. It is available in the folder “WinAC_ODK_Sample\STEP7_Prj\ODK_xamp.zip”.

Adapting the S7 program

You do not have to adapt the S7 program if you have used the suggested path during the installation of this example. If you have not used this path, please perform the following steps:

Table 5-7

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open the project BottleCheckS7 which you retrieved before.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Open DB652 contained in this project. DB652 contains all data required to initialize the ODK program.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Change to the declaration view of your Editor via View → Declaration view.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>The variable DLL_Name is at the first place in the DB. It contains the path to the library containing the ODK program (ODK_BottleChecker.dll). Change the path to where you installed the examples. The prefix “<strong>DLL:</strong>” has to be maintained. It indicates that it is a DLL (and not a COM object).</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Now change to the declaration view of your Editor via View → Declaration view.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>The data block has to be initialized to ensure that the actual value of the modified string corresponds to the initial value. To initialize the data block, click Edit → Initialize Data Block. The actual value now also includes the modified value.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Save the data block.</td>
<td></td>
</tr>
</tbody>
</table>
### Loading the program to WinAC RTX

The program can now be loaded to WinAC RTX.

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start WinAC RTX via Start→SIMATIC→PC Based Control→WinAC RTX</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Perform a memory reset of WinAC RTX by clicking the MRES button.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Change to STEP 7 and select the PC station PCWinAC in the opened project.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Open the PG/PC interface dialog box via Options→Set PG/PC Interface…</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Select “PC internal (local)” and exit the dialog box by clicking OK.</td>
<td><a href="image.jpg">Image</a></td>
</tr>
<tr>
<td>10</td>
<td>Load the blocks to the WinAC RTX controller by clicking <img src="image.jpg" alt="image" /></td>
<td></td>
</tr>
</tbody>
</table>
6 Operating the Application

Introduction

You will be provided with information on how to operate the application either via the variable table or via the visualization interface included in the delivery.

Starting the partial applications

Please follow the specified order, since correct functioning of the example is not ensured if this order is not observed.

Table 6-1

<table>
<thead>
<tr>
<th>No.</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set WinAC RTX to STOP by clicking the STOP button</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Start the variable table VAT_1 located in the block folder of the project included in the delivery. Click the &quot;Monitor Variable&quot; icon to display the values online.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Start the camera application &quot;VisionSensor_723.exe&quot; in the folder C:\WinAC_ODK_Sample\Cam_Application\Release&quot;</td>
<td>If you will use a vision sensor, make sure that the IP address corresponds to the IP address of your camera. If you won’t use a vision sensor, please select the “Simulate camera” option.</td>
</tr>
</tbody>
</table>
Part B: Installation of the Example Application

SIMATIC Vision Sensors and WinAC ODK

Operating the application

You can operate the application either from the variable table or via the interface of the camera application. If you operate on one side, this operation will become noticeable on the other side as a change. The ODK application does not have an individual user interface.

Operation via the variable table: Figure

The variable table has the following contents:

Figure 6-1

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Display</th>
<th>Status</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>//Variables for SetMode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MW  54 &quot;CommandID&quot;</td>
<td>HEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>M   301 &quot;SelMode_Play&quot;</td>
<td>BOCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>M   302 &quot;SelMode_Stop&quot;</td>
<td>BOCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>M   303 &quot;SelMode_Pause&quot;</td>
<td>BOCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>M   304 &quot;SelMode_Continue&quot;</td>
<td>BOCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>M   305 &quot;SelMode_WaterBottle&quot;</td>
<td>BOCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>M   306 &quot;SelMode_WineBottle&quot;</td>
<td>BOCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>//Acknowledgement of the above commands</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>M   307 &quot;AcknowledgmentBit&quot;</td>
<td>BOCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>MW  76 &quot;AckCommandID&quot;</td>
<td>HEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>MD  90 &quot;BottleCount&quot;</td>
<td>DEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>//Variables for GetMode (called every 20th OB1 cycle)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>//Combination 1 - Play, 2 - Stop, 3 - Pause</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>MW  70 &quot;CanMode&quot;</td>
<td>DEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>//Bottle type 1 = Water bottle, 2 = wine bottle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>MW  72 &quot;BottleType&quot;</td>
<td>DEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>//TestResult 0 = Passed, 1 = warning, 5 = none, -1 = Failed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>MW  74 &quot;LastTestResult&quot;</td>
<td>DEC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Press F1 for help.
Operation via the variable table: Meaning of the variables

The variables have the following meaning: R (read) mode represents variables which can only be monitored, W (write) mode represents variables which can and are to be actively modified by you.

Table 6-2

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command_ID</td>
<td>R</td>
<td>Contains a unique ID of the command which is transmitted (is automatically generated).</td>
</tr>
<tr>
<td>SetMode_Play</td>
<td>W</td>
<td>Setting camera application to Play mode. The running images of the vision sensor are displayed.</td>
</tr>
<tr>
<td>SetMode_Stop</td>
<td>W</td>
<td>Setting camera application to STOP mode. The running images of the vision sensor are no longer displayed and the inspection mode is terminated.</td>
</tr>
<tr>
<td>SetMode_Pause</td>
<td>W</td>
<td>Setting camera application to Pause mode. The running images of the vision sensor are displayed, but inspections are not performed.</td>
</tr>
<tr>
<td>SetMode_Continue</td>
<td>W</td>
<td>Canceling Pause mode and continuing inspections.</td>
</tr>
<tr>
<td>SetMode_WaterBottle</td>
<td>W</td>
<td>Selecting water bottle as test object.</td>
</tr>
<tr>
<td>SetMode_WineBottle</td>
<td>W</td>
<td>Selecting wine bottle as test object.</td>
</tr>
<tr>
<td>AcknowledgeBit</td>
<td>R</td>
<td>The ODK application has sent an acknowledgement to a command.</td>
</tr>
<tr>
<td>AckCommandID</td>
<td>R</td>
<td>Specifies the command (see Command_ID) for which the acknowledgement was sent via AcknowledgeBit.</td>
</tr>
<tr>
<td>BottleCount</td>
<td>R</td>
<td>A counter which is counted up by the vision sensor. Specifies the number of inspected bottles.</td>
</tr>
<tr>
<td>CamMode</td>
<td>R</td>
<td>Indicates the current mode of the camera application.</td>
</tr>
<tr>
<td>BottleType</td>
<td>R</td>
<td>Indicates which bottle type is being inspected.</td>
</tr>
<tr>
<td>LastTestResult</td>
<td>R</td>
<td>Displays the test result of the last bottle inspection.</td>
</tr>
</tbody>
</table>

Operation via the camera application: Figure

Instead of the variable table, the interface of the camera application can also be used to operate the application.
Part B: Installation of the Example Application

Overview

SIMATIC Vision Sensors and WinAC ODK

Figure 6-2

Operation via the camera application: Controls

Table 6-3

<table>
<thead>
<tr>
<th>Control</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit button</td>
<td>Terminates the application</td>
</tr>
<tr>
<td>Start button</td>
<td>Setting camera application to Play mode. The running images of the vision sensor are displayed.</td>
</tr>
<tr>
<td>Stop button</td>
<td>Setting camera application to STOP mode. The running images of the vision sensor are no longer displayed and the inspection mode is terminated.</td>
</tr>
<tr>
<td>Inspect Product:</td>
<td>By clicking this button, the bottle type to be inspected can be selected or the inspection can be paused by selecting the “Pause inspection” option.</td>
</tr>
<tr>
<td>Camera’s IP address</td>
<td>In this box, you enter the IP address of the vision sensor if you have connected a VS 723.</td>
</tr>
<tr>
<td>Bottle no</td>
<td>Displays the number of bottles which have already been inspected.</td>
</tr>
<tr>
<td>Test result</td>
<td>Displays the test result of the last bottle inspection.</td>
</tr>
<tr>
<td>Simulate Camera</td>
<td>Simulates the vision sensor. By selecting this option, you can start up the example without connecting a VS 723. The data which are otherwise delivered by the vision sensor (bottle counter, test result) are simulated.</td>
</tr>
</tbody>
</table>
Part C: Program Description

Overview

Contents of Part C

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

Objective of Part C

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

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

Requirement

For this part, it is helpful to have knowledge of SIMATIC as well as of Windows programming.

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

After providing you with information to understand the basic mechanisms of WinAC ODK, the Windows synchronization and data exchange mechanisms, we will now specifically show you how to usefully integrate them into individual automation solutions.

The control of an intelligent camera (SIMATIC VS 723 Vision Sensor) from WinAC RTX is used as a practical example. However, the mechanisms can be transferred to any other solution.

The specific example

To be able to explain the mechanisms of WinAC ODK and the data exchange between Windows applications using an example which is as specific as possible, we selected the bottle control as subtask in a bottling plant.

The bottle control is performed using a camera. It can independently differentiate between two bottle types, which were previously "trained". To ensure this, a command is communicated containing information on the type currently to be inspected. The arrival of a new bottle is directly reported to the camera via a sensor. The camera inspects the bottle and outputs the result which is then transferred to a controller.

Overview of the example application

The figure below provides an overview of the selected solution.

Figure 6-3
Elements of the selected solution

The solution consists of the following elements:

- **WinAC RTX**: Controls a plant (e.g. bottling plant); it requires data from the camera

- **WinAC ODK program**: Transfers the data from WinAC RTX to the camera application and vice versa. The basic framework of this application was created with the ODK wizard

- **Camera application**: Receives commands from WinAC RTX and controls the vision sensor; contains a very simple visualization of the images of the camera; here, the camera application stands in place of any Windows application

- **Shared memory**: Is used as data transfer medium between the two applications

- **Camera (vision sensor)**: Automatically inspects parts in the plant and reports the state of the part (good / warning / bad) to the camera application.

Communication of the elements involved

To ensure that WinAC RTX can perform its control tasks and that the vision sensor knows which bottle type is to be inspected, all elements have to be capable of exchanging data with each other and of synchronizing each other. To provide this, WinAC RTX sends commands to the vision sensor. In turn, the vision sensor sends test results to WinAC RTX.

Using this communication, we will also show you the application of the different CCX modes in WinAC ODK. You will also get to know the application of the different Windows mechanisms in practice.
7 Explanation of the Codes of the Individual Elements

Introduction
This chapter provides information on the most important core code sequences of the most important function elements. In individual cases, we refer to the commented code in the application.

Since the code in WinAC RTX and the code of the camera application are very closely connected to the code of the ODK application, these three elements are always explained in connection with each other.

Contents of the chapter Explanation of the STEP 7 Program

<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>56</td>
</tr>
<tr>
<td>7.1</td>
<td>Initializing WinAC ODK CCX</td>
<td>56</td>
</tr>
<tr>
<td>7.2</td>
<td>Example for a synchronous call of WinAC ODK CCX</td>
<td>58</td>
</tr>
<tr>
<td>7.3</td>
<td>Example for an asynchronous call of WinAC ODK CCX</td>
<td>63</td>
</tr>
<tr>
<td>7.4</td>
<td>Example for an ODK function which runs in the background</td>
<td>69</td>
</tr>
<tr>
<td>8</td>
<td>Adapting the Programs</td>
<td>74</td>
</tr>
<tr>
<td>8.1</td>
<td>Adapting the S7 program</td>
<td>74</td>
</tr>
<tr>
<td>8.2</td>
<td>Adapting the C++ applications</td>
<td>75</td>
</tr>
</tbody>
</table>

Introduction
We will show you how the functionalities were implemented in specific code, i.e.

- in WinAC RTX
- in the ODK application
- in the camera application

7.1 Initializing WinAC ODK CCX

Overview
Before the ODK application can be used by WinAC RTX, it is first required to start and initialize it. In our example, this is done by WinAC RTX in OB 100.

Description of the function
The initialization refers only to the S7 program in WinAC RTX and the ODK program.
Table 7-2

<table>
<thead>
<tr>
<th>Function component</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinAC RTX</td>
<td>In OB 100, WinAC RTX calls SFB 65001. This SFB starts and initializes the ODK program.</td>
</tr>
<tr>
<td>ODK application</td>
<td>In the ODK application, only the DLLMain () function is called which does not include any code.</td>
</tr>
</tbody>
</table>

Implementing the function

The relevant code locations for the two function components are shown in the following table.

Table 7-3

<table>
<thead>
<tr>
<th>Function component</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinAC RTX</td>
<td>OB100</td>
</tr>
<tr>
<td></td>
<td>CALL  SFB65001, DB65001</td>
</tr>
<tr>
<td></td>
<td>PROGID:=&quot;SFB 65002 DB&quot;.DLL_Name</td>
</tr>
<tr>
<td></td>
<td>STATUS:=&quot;SFB 65002 DB&quot;.Status</td>
</tr>
<tr>
<td></td>
<td>• ProgID contains path and file name (a DLL) of the ODK application</td>
</tr>
<tr>
<td></td>
<td>• Status contains either an error code or an identification number which has to be stored for later calls</td>
</tr>
<tr>
<td>ODK application</td>
<td>BOOL WINAPI DllMain</td>
</tr>
<tr>
<td></td>
<td>(HINSTANCE hinstDLL, // handle</td>
</tr>
<tr>
<td></td>
<td>DWORD fdwReason, // calling</td>
</tr>
<tr>
<td></td>
<td>reason</td>
</tr>
<tr>
<td></td>
<td>LPVOID lpvReserved // reserved</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>• The function is empty but it is expected by the operating system</td>
</tr>
</tbody>
</table>

Summary

Before using the ODK application for the first time, it has to be created and initialized by calling SFB 65001.
7.2 Example for a synchronous call of WinAC ODK CCX

Overview

A function for status check is implemented to receive information on the status of the camera application and the last test result of the bottle inspection from WinAC RTX.

The synchronous call is used if the functions of the ODK application can be processed very quickly (range: Few milliseconds). Synchronous calls block the OB1 cycle and may cause an exceeding of the maximum cycle time.

Description of the function

To ensure that data can be transferred from the camera to WinAC RTX, it is required that codes are implemented in all three function components.

Table 7-4

<table>
<thead>
<tr>
<th>Function component</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WinAC RTX</strong></td>
<td>1. First a unique command ID is generated in WinAC RTX. The purpose of this ID is to be able to uniquely identify a sent command.</td>
</tr>
<tr>
<td></td>
<td>2. WinAC RTX calls SFB 65002. A defined function in the ODK application is called. Parameters can be transferred during this call.</td>
</tr>
<tr>
<td></td>
<td>3. After the return of SFB 65002, the ODK application entered the desired data in the return parameters of the SFB. The command is called synchronous because the job is completed after the return of SFB 65002.</td>
</tr>
<tr>
<td><strong>Camera application</strong></td>
<td>4. A cyclic timer calls a timer function at regular intervals</td>
</tr>
<tr>
<td></td>
<td>5. In this timer function, the data are to be written into a shared area of the main memory. This requires that exclusive access to this memory is ensured. To provide this, a mutex is locked.</td>
</tr>
<tr>
<td></td>
<td>6. The data are written into the shared memory.</td>
</tr>
<tr>
<td></td>
<td>7. The mutex is released.</td>
</tr>
<tr>
<td><strong>ODK application</strong></td>
<td>1. The data requested by WinAC RTX are in the shared memory. The ODK application obtains exclusive access to this memory area by locking a mutex</td>
</tr>
<tr>
<td></td>
<td>2. The ODK application reads the desired data from the shared memory</td>
</tr>
<tr>
<td></td>
<td>3. The mutex is released.</td>
</tr>
</tbody>
</table>
Display of the data flow

The following figure shows the data flow for reading the camera status.

Figure 7-1

Display of the flow in a sequence diagram

The figure shows the sequence of a synchronous call.
Implementing the function

The relevant code locations for the three function components are shown in the following table.
### Part C: Program Description

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

SIMATIC Vision Sensors and WinAC ODK

Entry ID: 21572937

<table>
<thead>
<tr>
<th>Function component</th>
<th>Code</th>
</tr>
</thead>
</table>
| **WinAC RTX**       | // Create a unique command ID  
|                     | L "CommandID"  
|                     | + 1  
|                     | T "CommandID"  
|                     | // Call ODK  
|                     | CALL "EXEC_COM", DB65002  
|                     | OBJHandle :="SFB 65002 OB".Status  
|                     | Command :=DW#16#2  
|                     | InputData :="SFB 65002 OB".ODK Input  
|                     | OutputData :="SFB 65002 OB".ODK Output  
|                     | STATUS :="SFB 65002 OB".RetVal  
|                     | L "SFB 65002 OB".ODK_Output.Data1  
|                     | T "CamMode"  
|                     | L "SFB 65002 OB".ODK_Output.Data2  
|                     | T "BottleType"  
|                     | L "SFB 65002 OB".ODK_Output.Data3  
|                     | T "LastTestResult"  
|                     | • CommandID is the unique ID of the command  
|                     | • Command specifies the function to be executed in the ODK application (see below)  
|                     | • Input Data are input data for the ODK application  
|                     | • Output Data are output data of the ODK application  
| **Camera application** | dwMutexState = WaitForSingleObject  
|                     | (m_hSharedMemMutex, 800);  
|                     | // Set the current status to shared memory  
|                     | if (dwMutexState == WAIT_OBJECT_0)  
|                     | {  
|                     |     SharedMemStruct *smsMode;  
|                     |     smsMode = (SharedMemStruct *) m_pMapPtr;  
|                     |     smsMode->iBottleType = m_iCurrentBottleType;  
|                     |     smsMode->iLastTestResult = m_iTestResult;  
|                     |     smsMode->iMode = m_iCurrentMode;  
|                     | }  
|                     | ReleaseMutex (m_hSharedMemMutex);  
|                     | • WaitForSingleObjekt () locks the mutex  
|                     | • smsMode contains the pointer to the shared memory into which data are written  
|                     | • ReleaseMutex () releases the mutex |

--

V 1.1 Edition 05/17/05 61/79
Part C: Program Description
Explanation of the Codes of the Individual Elements

SIMATIC Vision Sensors and WinAC ODK

Entry ID: 21572937

ODK application

Function Execute:

```c
switch(command)
{
    case 1:  // Code for setting camera
        break;
    case 2:  // Example for a synchronous call
        retVal = GetCamMode(Input, Output);
        break;
}
```

- The yellow value specifies the function to be executed and is from the SFB 65002 call (see above)

Function GetCamMode:

```c
dwWaitResult = WaitForSingleObject
    (m_hSharedMemMutex, 5);
if (dwWaitResult == WAIT_OBJECT_0)
{
    Input.ODK_ReadS7INT (0, iCommandID);
    SharedMemStruct *pSharedMem = (SharedMemStruct*) m_pMapPtr;
    Output.ODK_WriteS7INT (0, iCommandID);
    Output.ODK_WriteS7INT (2,(ODK_SINT16) pSharedMem->iMode);
    Output.ODK_WriteS7INT (4, (ODK_SINT16) pSharedMem->iBottleType);
    Output.ODK_WriteS7INT (6,(ODK_SINT16) pSharedMem->iLastTestResult);
    result = ODK_SUCCESS;
}
```

- WaitForSingleObject locks a mutex
- pSharedMem contains a pointer to the shared memory
- WriteS7INT writes the output data which are read in SFB 65002 as OutputData

Summary

Using this code, the status of the camera can be transferred to the OutputData parameter of SFB65002 via the shared memory. This mechanism can be used for any Windows application which is to transfer data to WinAC RTX.
7.3 Example for an asynchronous call of WinAC ODK CCX

Overview

A function for setting the mode of the camera is implemented to change the mode of the camera from WinAC RTX.

Since this may last longer and thus cause an exceeding of the maximum cycle time in WinAC RTX, the function is implemented as asynchronous call. That means that the function in the ODK application runs simultaneously to the OB1 cycle. This is possible because Windows is a multitasking operating system.

Asynchronous functions are used if the processing of the function takes longer than an OB1 cycle or if the duration is unknown.

Description of the function

To ensure that the data sent by WinAC RTX are transferred to the camera, it is required that functionality is implemented in all three function components.

Table 7-6

<table>
<thead>
<tr>
<th>Function component</th>
<th>Functionality</th>
</tr>
</thead>
</table>
| **WinAC RTX**      | 1. First a unique command ID is generated in WinAC RTX. The purpose of this ID is to be able to uniquely identify a sent command.  
                   | 2. WinAC RTX calls SFB 65002 with respective parameters. A defined function in the ODK application is called.  
                   | 3. WinAC RTX makes available OB 52 which can be called directly from the ODK application.  
                   | The command is referred to as asynchronous because the job is still edited by the ODK application after the return of SFB 6502. |
| **ODK application**| 1. The ODK application reads and stores the input parameters of the SFB 65002 call. From these parameters, it detects that a command for setting the camera mode was called. That means that an asynchronous function has to be called.  
                   | 2. A new thread is started which now runs simultaneously to WinAC RTX.  
                   | 3. The thread sends a Windows message to the camera application.  
                   | 4. The thread waits for a Windows event coming from the camera application. It indicates the successful switching of the mode.  
                   | 5. The thread now calls OB 52 in WinAC RTX and thus informs the S7 program that the command was successfully executed. |
Camera application

1. The incoming Windows message causes the call of an associated function.
2. The mode of the camera application is set in the function. It may be required to send a command to the camera.
3. An event is set which signals the execution of the command to the ODK application.

Display of the data flow

The following figure shows the data flow for reading the camera status.

Figure 7-3
Display of the flow in a sequence diagram
The figure shows the sequence of an asynchronous call.

Implementing the function
The relevant code locations for the three function components are shown in the following table.
### Table 7-7

<table>
<thead>
<tr>
<th>Function component</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WinAC RTX</strong></td>
<td>Call of the asynchronous function</td>
</tr>
<tr>
<td></td>
<td>// Create a unique command ID</td>
</tr>
<tr>
<td></td>
<td>go: L &quot;CommandID&quot; + 1 T &quot;CommandID&quot;</td>
</tr>
<tr>
<td></td>
<td>T &quot;SFB 65002 DB&quot;.ODK_Input.CommandID</td>
</tr>
<tr>
<td></td>
<td>// Call ODK</td>
</tr>
<tr>
<td></td>
<td>CALL &quot;EXEC_COM&quot;, DB65002</td>
</tr>
<tr>
<td></td>
<td>OBJHandle :=&quot;SFB 65002 DB&quot;.Status</td>
</tr>
<tr>
<td></td>
<td>Command :=DW#16#1</td>
</tr>
<tr>
<td></td>
<td>InputData :=&quot;SFB 65002 DB&quot;.ODK_Input</td>
</tr>
<tr>
<td></td>
<td>OutputData :=P#DB652.DBX266.0 BYTE 4</td>
</tr>
<tr>
<td></td>
<td>STATUS :=&quot;SFB 65002 DB&quot;.RetVal</td>
</tr>
<tr>
<td>•</td>
<td>CommandID is the unique ID of the command</td>
</tr>
<tr>
<td>•</td>
<td>Command specifies the function to be executed in the ODK application (see below)</td>
</tr>
<tr>
<td>•</td>
<td>Input Data are input data for the ODK application</td>
</tr>
<tr>
<td>•</td>
<td>Output Data are output data of the ODK application</td>
</tr>
<tr>
<td><strong>Code of OB 52</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>// Command = SET_MODE_ACK??</td>
</tr>
<tr>
<td></td>
<td>L #Command</td>
</tr>
<tr>
<td></td>
<td>L W#16#20 ==I</td>
</tr>
<tr>
<td></td>
<td>JCN jer</td>
</tr>
<tr>
<td></td>
<td>L #DW2 T &quot;AckCommandID&quot;</td>
</tr>
<tr>
<td></td>
<td>SET S &quot;AcknowledgeBit&quot;</td>
</tr>
<tr>
<td></td>
<td>L 0 T &quot;OB52ErrCode&quot;</td>
</tr>
<tr>
<td></td>
<td>JU go</td>
</tr>
<tr>
<td>// Command = Unknown Command</td>
<td></td>
</tr>
<tr>
<td>jer: L #Command T &quot;OB52ErrCode&quot;</td>
<td></td>
</tr>
</tbody>
</table>
Part C: Program Description
Explanation of the Codes of the Individual Elements

SIMATIC Vision Sensors and WinAC ODK
Entry ID: 21572937

- The parameters which were specified in the ODK application can be polled via the local data of OB 52. It was determined arbitrarily that the value 20h in the local date “Command” means the SET_MODE_ACK function.
- Other local data are transferred to memory words and are available for evaluation by other S7 program parts.
- In case of an unknown command, an error code is stored in a memory word.

<table>
<thead>
<tr>
<th>Camera application</th>
<th>Messages jump distributor:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN_MESSAGE_MAP(CVisionSensor_723Dlg, CDialog)</td>
<td></td>
</tr>
<tr>
<td>ON_MESSAGE(WM_SET_MODE, SetMode)</td>
<td></td>
</tr>
<tr>
<td>END_MESSAGE_MAP()</td>
<td></td>
</tr>
</tbody>
</table>

- Defines that the Windows message WM_SET_MODE causes a call of the SetMode function

```cpp
dwMutexState = WaitForSingleObject  
    (m_hSharedMemMutex, 800);

// Set the current status to shared memory
if (dwMutexState == WAIT_OBJECT_0)
{
    SharedMemStruct *smsMode;  
    smsMode = (SharedMemStruct *) m_pMapPtr;  
    smsMode->iBottleType = m_iCurrentBottleType;  
    smsMode->iLastTestResult = m_iTestResult;  
    smsMode->iMode = m_iCurrentMode;
}
ReleaseMutex (m_hSharedMemMutex);
```

- WaitForSingleObjekt () locks the mutex
- smsMode contains the pointer to the shared memory into which data are written
- ReleaseMutex () releases the mutex
### ODK application

#### Function Execute:

```c
switch(command) {
    case 1:
      // Example for an asynchronous thread
      retVal = SetCamMode(Input, Output, hBottleCheckApp);
      break;
    case 2:
      // Code for getting Cam Mode
      break;
}
```

- The yellow value specifies the function to be executed and is from the SFB 65002 call (see above)

#### Function SetCamMode:

```c
asyncSetMode *SetModeEvent = new (asyncSetMode);
Input.ODK_ReadS7INT (0, SetModeEvent->m_iCommandID);
Input.ODK_ReadS7INT (2, SetModeEvent->m_iMode);
Input.ODK_ReadS7INT (4, SetModeEvent->m_iData);
// Pass the window handle
SetModeEvent->m_hBottleCheckWin = hBottleCheckApp;
SetModeEvent->m_hSvcHandle = g_ServiceHandle;
// Create the asynchronous thread by sending an event
g_Processor->ScheduleEvent (SetModeEvent);
```

- ODK_Read... Functions read the input parameters from the SFB 65002 call
- SetModeEvent is a class of which the instance can be executed as simultaneous thread.
- ScheduleEvent starts the execution of the thread
Part C: Program Description
Explanation of the Codes of the Individual Elements

SIMATIC Vision Sensors and WinAC ODK
Entry ID: 21572937

Function asyncSetMode::Execute()

SendMessage (m_hBottleCheckWin, WM_SET_MODE,
(WPARAM) m_iMode,
(LPARAM) m_iData);

// Wait for event that signals acknowledge
if (m_hSetModeEvent != NULL)
// return after timeout = 1 ms
    dwWaitResult = WaitForSingleObject
        (m_hSetModeEvent, 1);
if (dwWaitResult == WAIT_OBJECT_0)
// Signaled, set mode was successful
// Notify WinAC, that SetMode was successful
    by calling OB 52
        ODK_ScheduleOB (m_hSvcHandle, 0x11, 0x41,
            0xFE, 52, 0xC4, 0x59, SET_MODE_ACK,
            m_iCommandID);

ResetEvent (m_hSetModeEvent);

- SendMessage sends the Windows message to the camera application
- WaitForSingleObject waits for a Windows event from the camera application; a timeout is also specified
- ODK_ScheduleOB causes that the specified OB (here: OB 52) is executed in WinAC RTX
- ResetEvent resets the event

Summary

You could see the interaction of most different Windows and ODK mechanisms:

- Asynchronous call of ODK in WinAC RTX
- Data transfer from and to the ODK application
- Windows messages
- Windows events
- Call of OBs in WinAC RTX

7.4 Example for an ODK function which runs in the background

Overview

Some tasks require to continuously perform checks which are executed independent of the other automation tasks (polling). This can be solved with continuous functions of WinAC ODK CCX.

Continuous functions are used if tasks have to be cyclically processed similarly to OB1. To ensure this, an individual thread is created.
Description of the function

The specific example monitors a counter variable located in the shared memory. The counter represents the number of inspected bottles. Whenever this counter increases, the value is to be written into a variable in WinAC RTX.

<table>
<thead>
<tr>
<th>Function component</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinAC RTX</td>
<td>It is not required to write a code in WinAC RTX.</td>
</tr>
<tr>
<td>Camera application</td>
<td>After each completed inspection, the camera application writes the number of inspected bottles into the shared memory.</td>
</tr>
<tr>
<td>ODK application</td>
<td>1. In the continuous function (a task which, similarly to OB1, runs cyclically in the ODK application), a counter variable is monitored in the shared memory (polling).&lt;br&gt;2. If the counter variable is modified, the ODK_WriteData (...) function is called which writes the value of the counter variable directly into a memory double word of WinAC RTX.</td>
</tr>
</tbody>
</table>

Display of the data flow

The following figure shows the data flow for writing the number of inspected bottles into WinAC RTX.

Figure 7-5
Display of the flow in a sequence diagram

The figure below shows the sequence for writing the number of inspected bottles into WinAC RTX.

Implementing the function

The relevant code locations for the three function components are shown in the following table.

<table>
<thead>
<tr>
<th>Function component</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera application</td>
<td>Callback function of the camera:</td>
</tr>
<tr>
<td></td>
<td>// Write bottle number into shared memory</td>
</tr>
<tr>
<td></td>
<td>pSharedMem-&gt;ulBottleNo = (unsigned long)dImageNo;</td>
</tr>
<tr>
<td></td>
<td>• The camera features an internal counter in which it counts the</td>
</tr>
<tr>
<td></td>
<td>number of inspected bottles. After each completed inspection,</td>
</tr>
<tr>
<td></td>
<td>this value is written into the variable dImageNo via a callback</td>
</tr>
<tr>
<td></td>
<td>function.</td>
</tr>
<tr>
<td></td>
<td>• The number is written into the shared memory (pSharedMem)</td>
</tr>
<tr>
<td></td>
<td>with the above function.</td>
</tr>
</tbody>
</table>
### Function `BottleCheckerMonitor::Execute()`

```cpp
newBottleNo = (unsigned long) pSharedMem->ulBottleNo;
if (newBottleNo != m_ulBottleNo)
// New bottle count! Send to WinAC
{
    ODK_DATA_STRUCT dsStruct;
    ODK_BIT32 WriteVal;
    ODK_RESULT ret;

    dsStruct.dataType = ODK_DATA_TYPE_DWORD;  // Access double word
    dsStruct.quantity = 1;                     // Write 1 double word
    dsStruct.dbNumber = 0;                    // DB Number, not relevant
    dsStruct.memoryArea = ODK_MEM_AREA_M;      // Access flag area
    dsStruct.areaOffset = 90;                  // Write MD 90 in WinAC
    dsStruct.bitNumber = 0;                    // Bit number, not relevant
    dsStruct.pBuff = (unsigned char*) &WriteVal; // Set pointer to data
    dsStruct.maxSize = sizeof(WriteVal);       // Set max size of data = 1 DWORD
    dsStruct.status = 0;                       // Return Value: Status

    // Use S7 access methods for byte swapping
    CWinACReadWriteData readSwap
    (dsStruct.maxSize, dsStruct.pBuff);

    WriteVal = newBottleNo;

    // Use S7 access methods for byte swapping
    ret = readSwap.ODK_WriteS7DWORD(0, WriteVal); // Write data
    ret = ODK_WriteData(m_WinACSvc, 1, &dsStruct, true);
}
```

- The number of inspected bottles is read from the shared memory into the variable `newBottleNo`
- If the number has changed, this fact first has to be reported to WinAC RTX
- To report the change, a structure is filled out which contains information on the variable to be written in WinAC RTX. Via the above structure, the memory double word 90 is initialized.
Part C: Program Description

Explanation of the Codes of the Individual Elements

SIMATIC Vision Sensors and WinAC ODK

<table>
<thead>
<tr>
<th>WinAC RTX</th>
<th>• A code is not required in WinAC RTX. It is written directly into the variable MD 90.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Since the arrangement of the bytes in doublewords in the PC is different from the arrangement in a SIMATIC controller, it is first required to bring them into the correct order using the ODS_WriteS7DWORD function.</td>
<td></td>
</tr>
<tr>
<td>• ODK_WriteData writes the data directly into WinAC RTX</td>
<td></td>
</tr>
</tbody>
</table>

Summary

Using the ODK_WriteData function, data can be directly written from Windows applications into the different memory areas of the WinAC RTX controller. Functions for cyclic writing and reading as well as for simple reading from WinAC RTX are also available.
8 Adapting the Programs

Introduction

This chapter provides information on how to adapt the individual elements of this application to your specific requirements. This application is only an example for a possible integration of a Windows application into WinAC RTX control programs.

Contents of this chapter

Table 8-1

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Adapting the Programs</td>
<td>74</td>
</tr>
<tr>
<td>8.1</td>
<td>Adapting the S7 program</td>
<td>74</td>
</tr>
<tr>
<td>8.2</td>
<td>Adapting the C++ applications</td>
<td>75</td>
</tr>
</tbody>
</table>

8.1 Adapting the S7 program

This example

The S7 program included in the delivery of this application shows only three important points:

- Initializing the ODK program
- Calling functions in the ODK program
- Implementing OB 52 which is available for calls from the ODK application

The S7 program contains no control functionality whatsoever for the direct control of external SIMATIC I/O.

Adaptations for individual applications

For your individual application, implement the control functionality for your system in the S7 program. External I/O can be addressed via PROFIBUS CPs or via the integrated Ethernet interface. The examples on PC-based automation listed above are available in the application portal of A&D Service & Support.

You can call your individual (synchronous, asynchronous or continuous) functions in the ODK application according to the code sequence examples shown above.
8.2 Adapting the C++ applications

Variants for your application

In the C++ applications, you implement Windows functionalities which are to be integrated into your control program. Two options are available:

- Implementing the functionality in an individual Windows application which communicates with the ODK application
- Implementing the functionality directly in the ODK application

8.2.1 Adaptations in the Windows application

This example

The purpose of the camera application included in the delivery of this application is merely to illustrate how a Windows application can communicate with the ODK application. The following mechanisms are used:

- Windows messages
- Shared memory (commonly used main memory)
- Mutexes (object for the locking of Windows resources)
- Events (notification object)

This program is very specifically designed for the task defined at the beginning and can probably not be used for your individual requirements. It is only an example of the communication mechanisms. You can use the program to see how such mechanisms can be implemented in Windows.
Graphic display of a general solution

Adaptations for your individual application

Usually you will completely recreate the Windows program. It is of course possible to use an existing source code. However, you have to expand it by the communication and synchronization mechanisms listed above via which you establish a connection to the ODK application.

8.2.2 Adaptations in the ODK application

This example

The ODK application included in the delivery of this application is only used as a carrier of data from WinAC RTX to the camera application and vice versa. It does not contain an individual control functionality. The following is shown:

- Synchronous functions of the ODK application
- Asynchronous functions of the ODK application
- Continuous functions in the ODK application
Graphic display of a general solution

Figure 8-2

Adaptations for your individual application

You can also program your functionality directly in the ODK application. The advantage is the fact that you do not have to consider the communication and synchronization with the Windows application. It is also recommended if large data amounts have to be quickly processed.

This solution is not suitable if the application is to feature a visible Windows window. Implementing windows in WinAC ODK requires more effort.

Examples which can be realized very cleverly that way are:

- Controllers or other mathematical algorithms in C++
- Interfacing of databases
- Integration of drivers for PC expansion cards of third-party manufacturers
- Sending e-mails

The ODK wizard supports you during creating the basic framework of the application.
# Bibliographic References

This list is by no means complete and only provides a selection of appropriate sources. After installing the respective product, most manuals are available via:

Start → Simatic → Documentation → English

The Product Support is available on the internet at:

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>Description of functions and operation of WinAC RTX V4.1.</td>
<td>SIMATIC WinAC RTX V4.1</td>
</tr>
<tr>
<td></td>
<td>Available on the WinAC RTX V4.1 CD or directly from the control panel: Menu Help → Help on Controller.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Description or information on:</td>
<td>SIMATIC NET Commissioning PC</td>
</tr>
<tr>
<td></td>
<td>• General information on the PC tools</td>
<td>Stations – Instruction and Quick Start for SIMATIC NCM PC / STEP 7 version V5.2 and higher.</td>
</tr>
<tr>
<td></td>
<td>• Functions of NCM PC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Available in the Product Support; Entry ID: 13542666</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Manual for industrial communication on PG/PC with SIMATIC NET.</td>
<td>SIMATIC NET – Industrial Communication with PG/PC</td>
</tr>
<tr>
<td></td>
<td>Available in the Product Support; Entry ID: 16923753</td>
<td></td>
</tr>
<tr>
<td>4</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. Available in the Product Support; Entry ID: 1214574</td>
<td>System Software for S7-300/400 System and Standard functions</td>
</tr>
<tr>
<td>5</td>
<td>SIMATIC Rack PC IL40S manual</td>
<td>SIMATIC Rack PC IL40S manual</td>
</tr>
<tr>
<td></td>
<td>Available in the Product Support; Entry ID: 15317654</td>
<td></td>
</tr>
</tbody>
</table>

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