WinAC

Link to an SQL Database to WinAC RTX

User documentation

V1.20 • November 2009

Applikationen & Tools

Answers for industry.

SIEMENS

Industry Automation and Drives Technologies Service & Support Portal

This article is taken from the Service Portal of Siemens AG, Industry Automation and Drives Technologies. The following link takes you directly to the download page of this document.

http://support.automation.siemens.com/WW/view/en/48354880

If you have any questions concerning this document please e-mail us to the following address:

online-support.automation@siemens.com

SIEMENS

SIMATIC
WinAC Link to an SQL Database

Basic Information 1

Overview 2

Installation MySQL 3

Installation MsSQL 4

Functional Description 5

Detailed Description of FBs 6

Application Examples 7

Error Messages 8

List of Abbreviations 9

History 10

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These application examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice.
If there are any deviations between the recommendations provided in these application examples and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act ("Produkthaftungsgesetz"), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract ("wesentliche Vertragspflichten"). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of Siemens Industry Sector.

Table of Contents

Warranty and Liability **4**

Instruction **7**

1 Basic information **8**

1.1 Objective 8
1.2 Required Expertise 8
1.3 Reference System 8

2 Overview **9**

2.1 Function Scope 9
2.2 Version of the Driver 9

3 Installation MySQL **11**

3.1 Quickstart with Example Project 11
3.2 Installation of MySQL Server and Recovery of Schematic 11
3.2.1 Installation of MySQL Database 11
3.2.2 Recovery of Example Schematic 14
3.2.3 Installation of GUI-Tools 15
3.2.4 Define a User 17
3.2.5 Transaction Safe Data Transmission to MySQL-Server 19
3.3 Installation WinAC Driver on Runtime Computer 19
3.3.1 Installation of DLL 19
3.3.2 Installation and Parameterisation of the ODBC Driver 19
3.4 Installation WinAC Driver on SIMATIC Engineering Computer 22

4 Installation MsSQL **23**

4.1 Quickstart with Example Project 23
4.2 Installation of MsSQL Server and Recovery of the Schematics 23
4.2.1 Installation of MsSQL Database 23
4.2.2 Installation of Microsoft SQL Server Management Studio Express ... 26
4.2.3 Recovery of Example Schematic using the SQL Server Management Studio Express 26
4.2.4 Important Server Settings 27
4.2.5 Create a Logon to the SQL Server 29
4.3 Installation WinAC Driver on Runtime Computer 31
4.3.1 Installation of DLL 31
4.3.2 Installation und Parameterisation of ODBC Driver 31
4.4 Installation WinAC Driver on SIMATIC Engineering Computer 36

5 Functional Description **37**

5.1 Basics 37

6 Detailed Description of FBs **39**

6.1 ODK Initialisation Block 39
6.2 ODBC Communication Build-up to the SQL Datenbase 42
6.3 ODBC Read and Write Block for SQL Database 46
6.4 ODBC Block for Closing the Communication 53

7 Application Examples **57**

7.1 The Use of the STEP 7 Example Project 57
7.1.1 Structure of an Application Programme 57
7.1.2 Sending of a SELECT Statement using the variable table “Control “ 58
7.2 Adaptation of STEP 7 Example to User’s Own Requirements 58
7.2.1 Other SQL statements than in example project 58

8 Error Messages **59**

8.1 Error Messages of WinAC ODK 4.1 59
8.1.1 Error Messages for SFB65001 → STATUS_ODK_CREA_CON..... 59
8.1.2 Error Messages for SFB65002 STATUS_ODK_CON/EXEC..... 60
8.2 Special Error Messages of SQL-DB Driver 60
8.2.1 ODK-Function Returns 60
8.2.2 Function-Code Numbers 62
8.2.3 ODBC-Function Errors 62
8.2.4 SQL-Statements 63

9 List of Abbreviations 67

10 History..... 68

Instruction

Content

This document describes the WinAC driver for an SQL database link to WinAC RTX via an ODBC driver.

1 Basic information

1.1 Objective

Develop of a WinAC RTX driver for a direct connection to a SQL database.

Four SQL statements (SELECT, UPDATE, INSERT, DELETE) can be transmitted in any reasonable order.

This document describes how, by means of this driver, SQL statements can be transmitted to the SQL server, and how data can be collected.

Note Basically the driver can be used for databases with an ODBC driver.

1.2 Required Expertise

In order to understand this document you need to be conversant with the following documents:

Table 1-1 Required expertise

System	Document
STEP 7	S7prv54_d.pdf
MsSQL	http://technet.microsoft.com/de-de/library/ms165706.aspx
MySQL	DokuMySQL-5.1-de.a4.pdf

1.3 Reference System

- SIMATIC Microbox PC 427B (1 GHz, 512 MB RAM, 1 GB Flash) with Windows XP embedded SP2
- WinAC RTX 2009
- SIMATIC NET V6.3 + HF1
- STEP 7 V5.4 + SP4
- SQL-Server odbc 03.85.1117
- Microsoft SQL 2005 Server Express Edition Version 9.00.1399.06.06
- SQL Server Management Studio Express
- mysql-connector-odbc-3.51.14-win32
- mysql-5.0.37-win32
- mysql-gui-tools-5.0-r11a-win32

2 Overview

2.1 Function Scope

The following functions of the SQL data link are supported by means of this driver:

- to transmit SQL statements with flexible arguments
  possible statements:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
- to receive read data from the SQL database

2.2 Version of the Driver

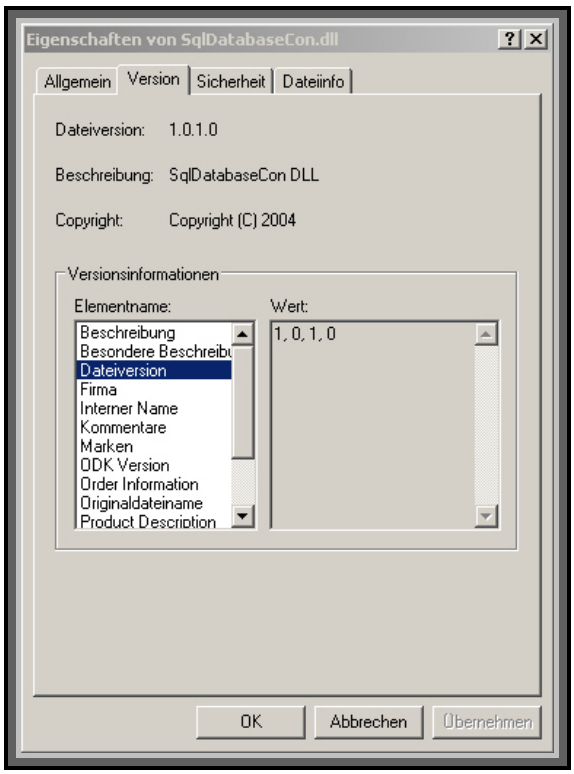
Determine Driver Version Under Windows

The registered driver DLL is located in the system directory, e.g.

C:\WINNT\system32\dll or C:\Windows\system32\dll

Determine the driver DLL version by viewing the file properties in Windows Explorer (right mouse click → properties).

Figure 2-1 Version of Driver DLL



Determine Driver Version in STEP 7 Project

Determine the version of the STEP 7 blocks in the instance DB of SQL_CON (DBI_SQL_CON):

C_IF.STEP 7_VERSION	Version of STEP 7driver SW

3 Installation MySQL

3.1 Quickstart with Example Project

- Database Server
  - Install database system MySQL on database server
  - Install mysql-gui-tools
  - Recover Scales Machine Schematic in MySQL database
  - Create an SQL server logon
- Runtime System
  - Install DLL with the DLL_Install.bat file on Runtime System
  - Install MyODBC driver on Runtime System
  - Parameterise ODBC driver installed on Runtime System on Scales Machine database
- SIMATIC Engineering Computer
  - De-archive MySQL_BspPrj STEP 7 project on SIMATIC Engineering computer
  - Adapt ODBC connection data in DB10
  - Transfer STEP 7 project in WinAC RTX (Runtime System)

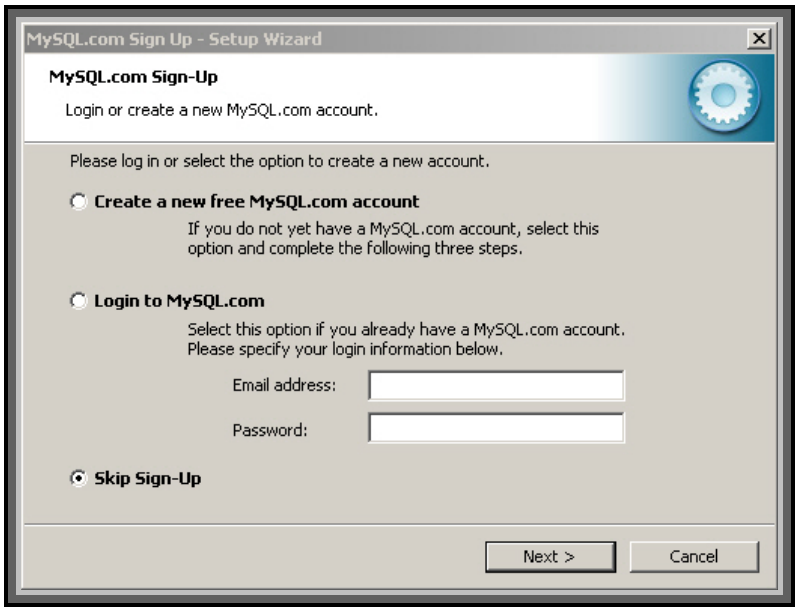
Optionally it is possible to install MySQL-5.0-r11a-win32.

3.2 Installation of MySQL Server and Recovery of Schematic

3.2.1 Installation of MySQL Database

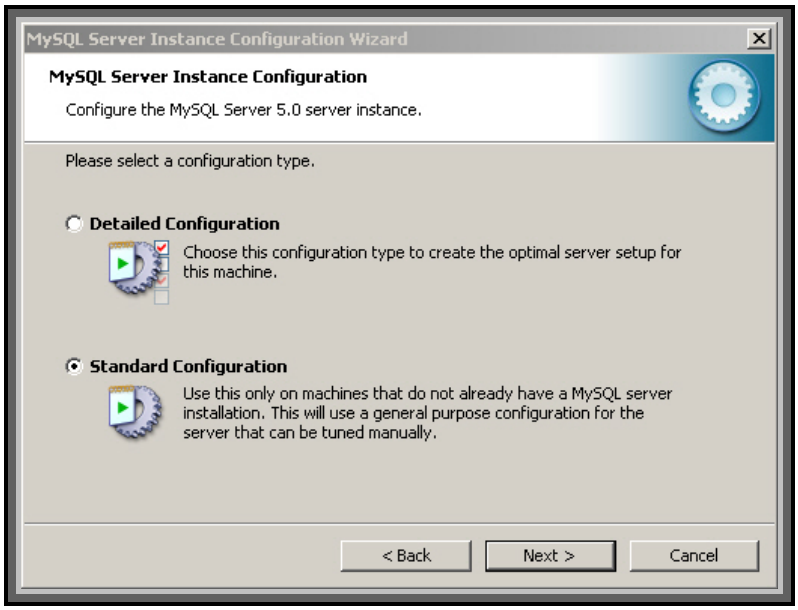
The database system “mysql-5.0.37-win32” is installed on the database server. In order to install the database, unpack the packed file “mysql-5.0.37-win32.zip“ and run Setup.exe. During the installation, select one of the setup types “Typical“ or “Complete“. After the installation it is decided whether a MySQL account should be generated. Select “Skip Sign-Up“.

Figure 3-1 Skip MySQL-Account



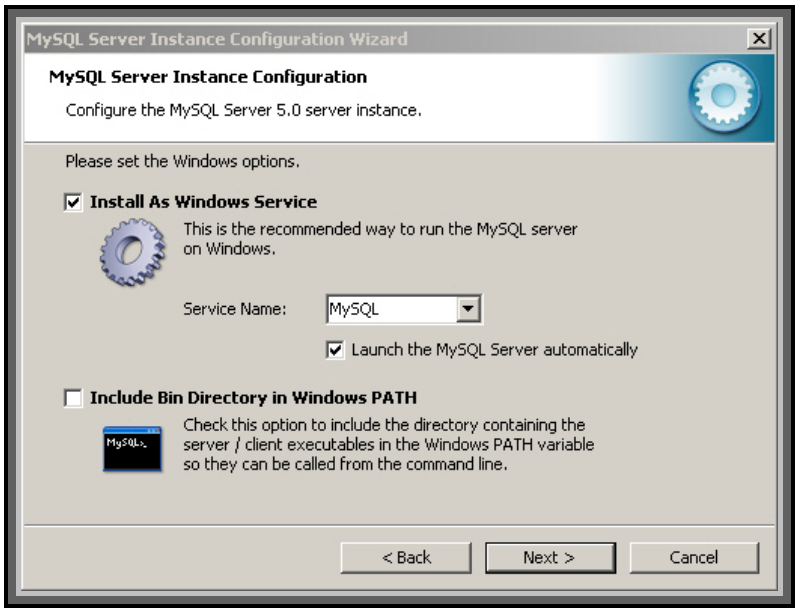
This is followed by the configuration of the MySQL server. For the application example select the standard configuration.

Figure 3-2 MySQL Configuration



This is followed by the server instance configuration. The appropriate settings for the example application are shown in the figure below.

Figure 3-3 Server Instance Configurations



Refer to the next picture for the assignment of a root password for the server.

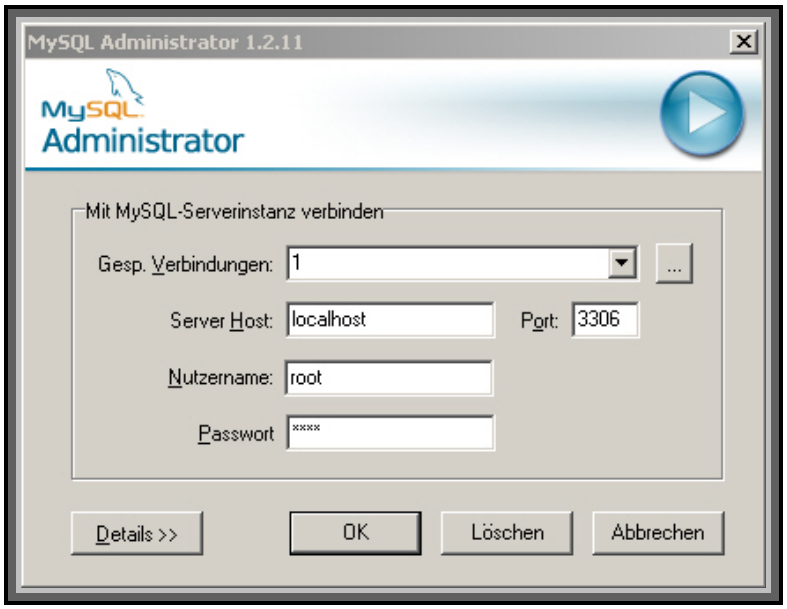
The next step is the execution of the completed configurations by the database.

After the installation a MySQL Command Line Client is made available in order to transfer the database administrations and SQL statements. Use the MySQL GUI Tools if graphic administrations such as create and manage databases are required (see installation on SIMATIC Engineering computer).

3.2.2 Recovery of Example Schematic

Execute the MySQL Administrator in order to re-insert the saved project into the database. In order to create a connection to the MySQL-Server, first the login data are required. Please refer to the following picture for an illustration of a “localhost“ connection.

Figure 3-4 MySQL Administrator localhost connection

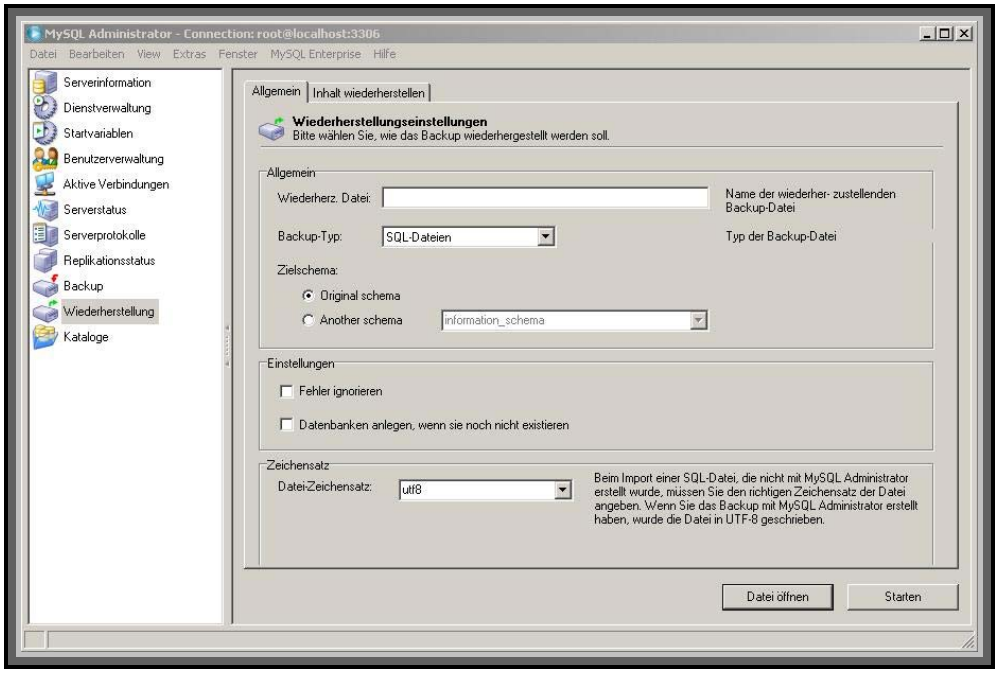


Refer to the next picture for an illustration of a connection via TCP/IP. Please note that certain administrations are only possible via “localhost“.

Figure 3-5 MySQL Administrator TCP/IP connection

The next step is a window for database administration. Use the appropriate tab page for recovery. Then use “Open File“ to select the file “Scales_MachineV1.0.sql“ from the example project. Recovery is initiated by clicking “Start“.

Figure 3-6 MySQL Recovery



3.2.3 Installation of GUI-Tools

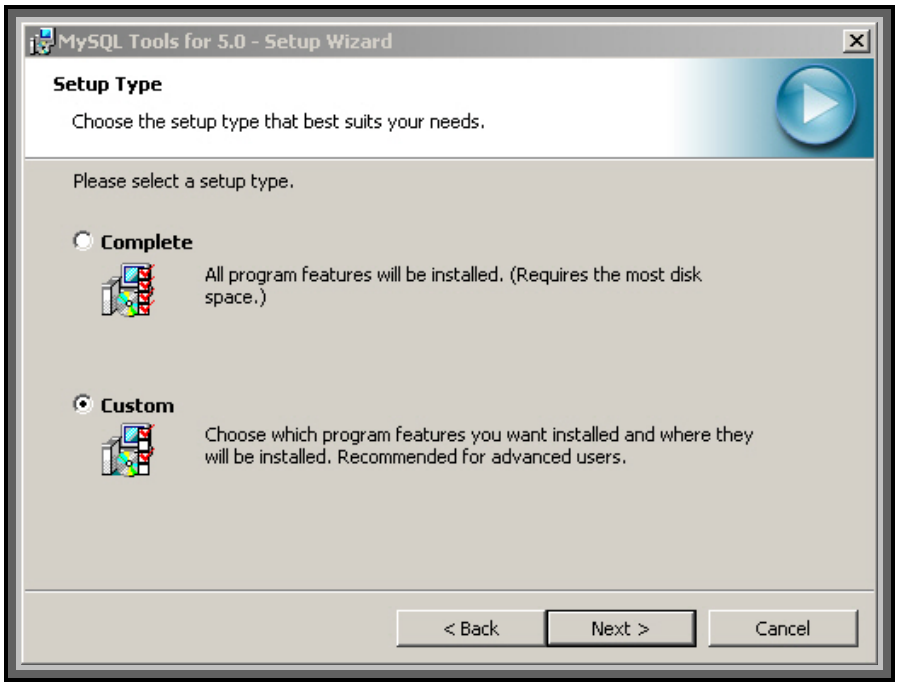
To ensure you can use the example project it is required to install additional administration tools of the MySQL database. For this purpose you need to execute the file “mysql-gui-toolsMySQL-5.0-r11a-win32.msi“. Once installation is complete the following four programmes for the database administration are available: -

- MySQL Administrator
- MySQL Migration Toolkit
- MySQL Query Browser
- MySQL System Tray Monitor

The MySQL Administrator serves to carry out the administration of the database. Use MySQL Migration Toolkit to migrate various schematics and data into MySQL databases.
Amongst other things the browser is for the graphic illustration of data in a table. With the System Tray Monitor you can display various runtime events and switch the server instance on and off.

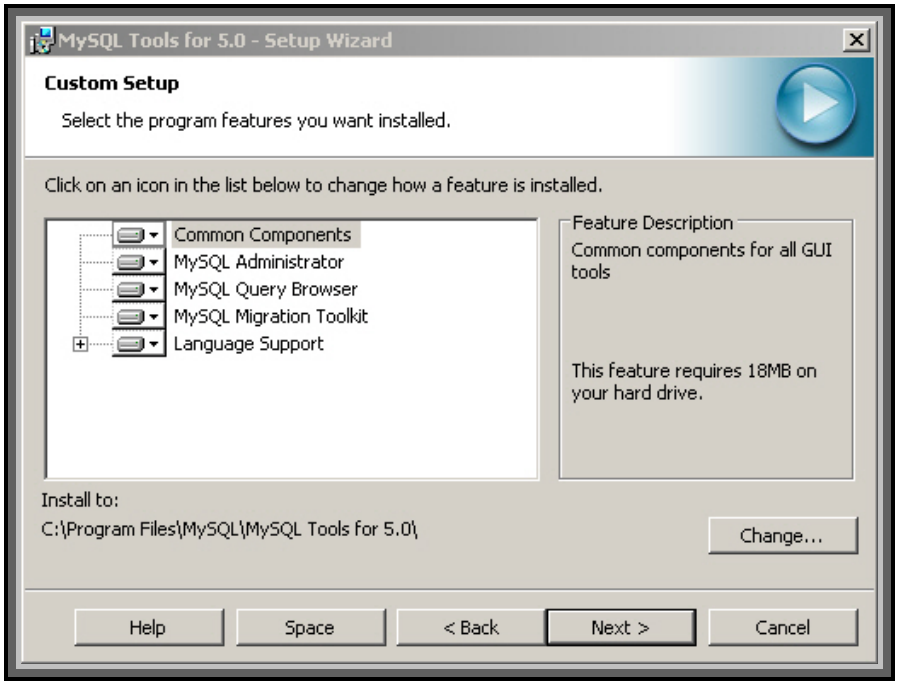
In order to install the GUI-Tools execute the file “mysql-gui-toolsMySQL-5.0-r11a-win32.msi“. After selecting the installation path, you can choose either a complete installation or a user defined one.

Figure 3-7 Choose setup type



When the user defined installation has been chosen, certain tools can be deselected. It is, however, recommended to install all tools.

Figure 3-8 GUI Components



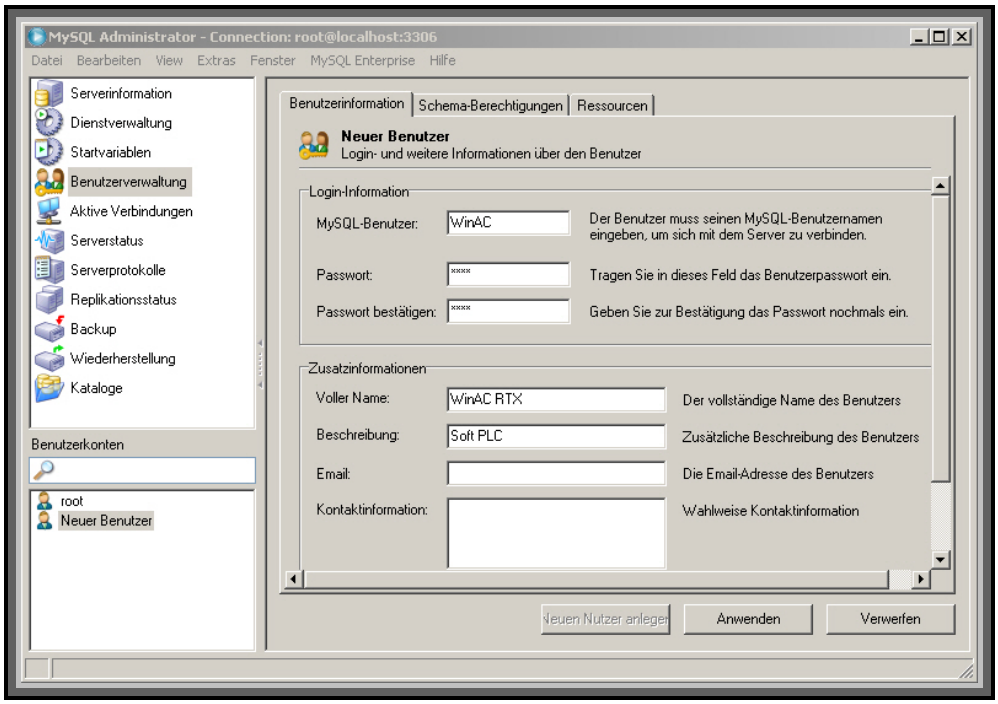
Once installation has been completed it is recommended to put the MySQL System Tray Monitor in Autostart to enable you to start and stop the database via the icon in the task bar.

3.2.4 Define a User

To establish a connection to the database via TCP/IP it is required to define a user with assigned authorizations. Prior to being able to define a user first you need to have installed the MySQL-GUI Tools on the development environment (see Chapter 3.2.3).

First start the MySQL Administrator. The key “define new user” under user administration enables you to define a new user.

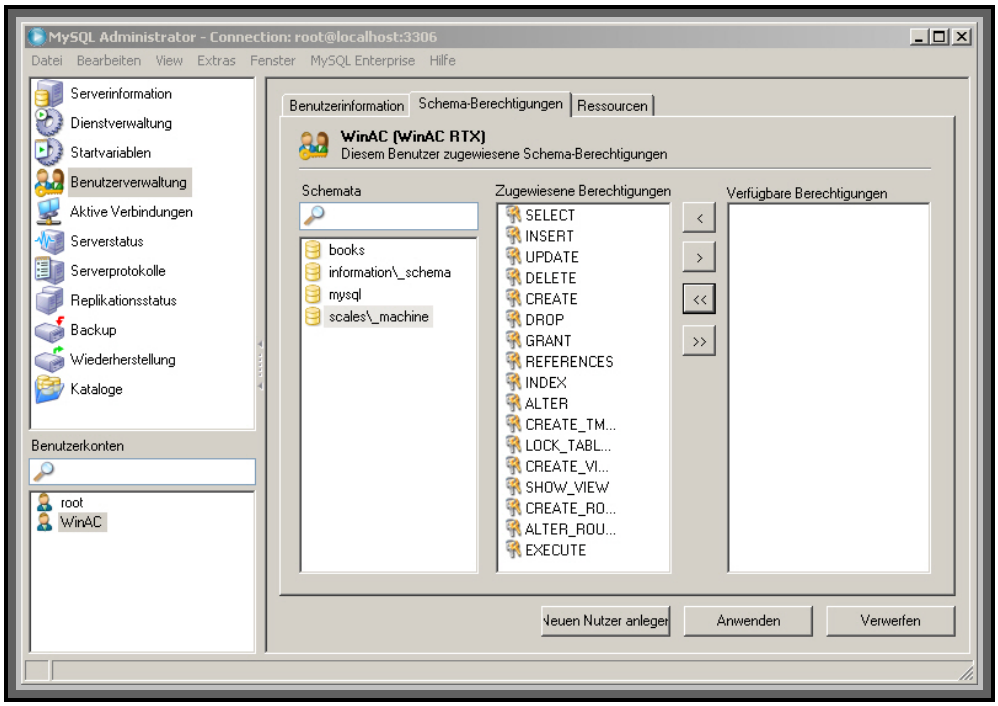
Figure 3- 9 Define user



Use the Apply key to save the user in the database.

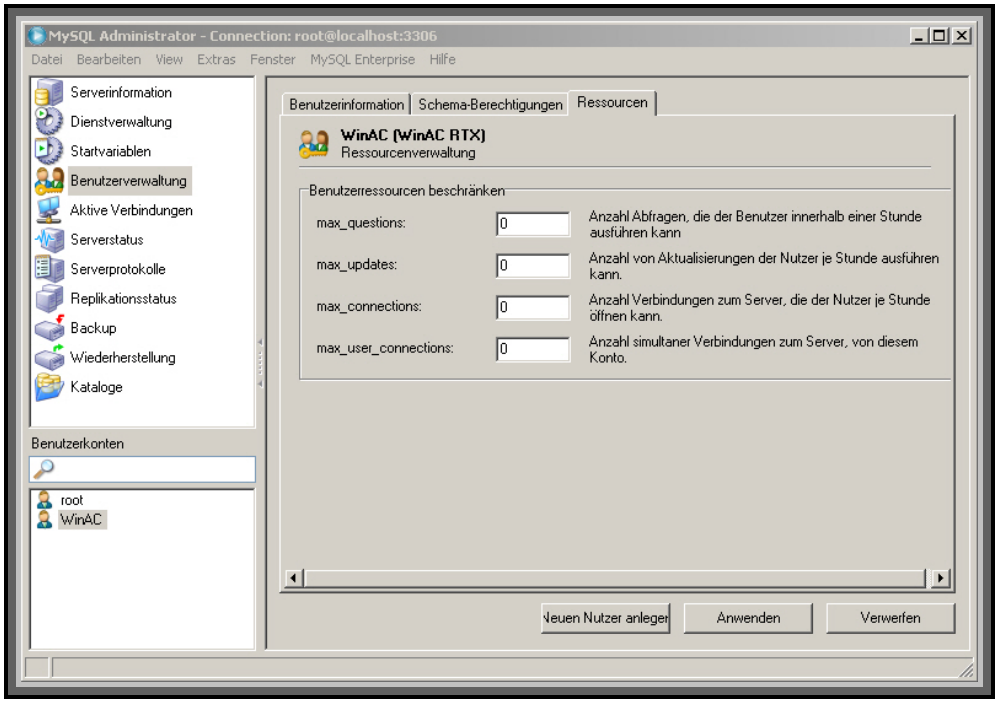
The access authorisations for the respective user are parameterised under the tab page Schematics Authorizations.

Figure 3-10 Schematics Authorizations



User resources can be limited under the tab page Resources.

Figure 3-11 User resources



3.2.5 Transaction Safe Data Transmission to MySQL-Server

MySQL runs in Autocommit mode as standard. This means that as soon as a statement is processed which updates (i.e. changes) a table, MySQL saves this change on the hard disk.

If, however, it is required to work with transaction safe data transmission and off commands, please refer to Chapter “Transactional and Off Commands of MySQL“ in the MySQL documentation.

3.3 Installation WinAC Driver on Runtime Computer

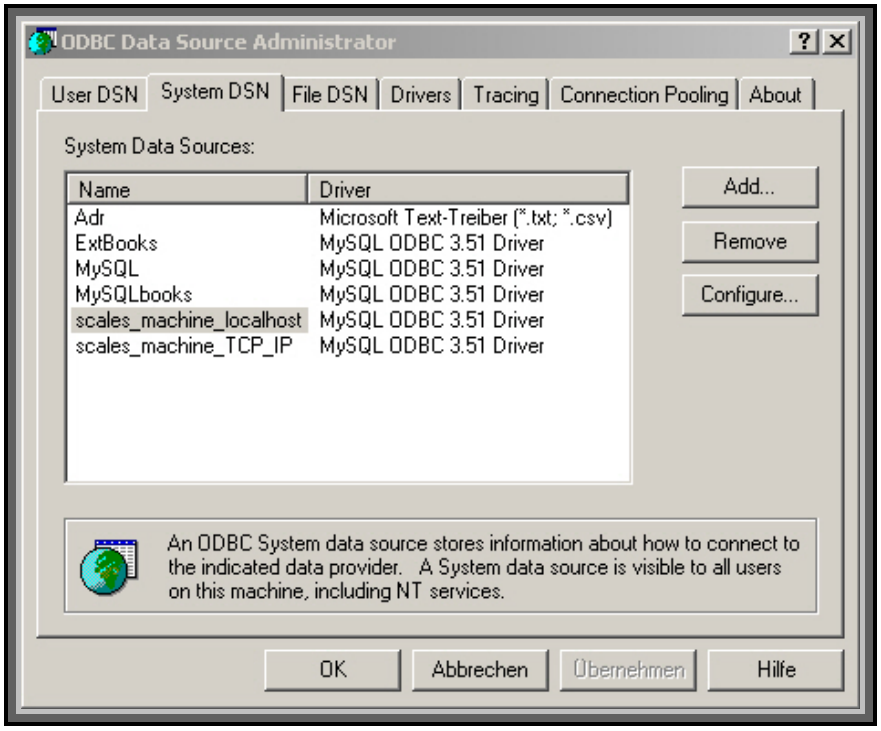
3.3.1 Installation of DLL

The installation of the WinAC driver for the SQL database connection is limited to the execution of the DLL-Install.bat. During the installation the DLL must be located in the same directory as the bat files.
In addition the MyODBC driver must be installed and parameterised.

3.3.2 Installation and Parameterisation of the ODBC Driver

The ODBC driver is used to establish the connection to the SQL database. The driver is installed by means of the “mysql-connector-odbc-3.51.14-win32“ installation file. During the installation you choose between the setup types “Typical“ or “Complete“. After the installation of the MyODBC-driver has been completed, it is required to parameterise the database interface. The ODBC administration is located under s*ystem control\administration\data sources (ODBC)*. This is located on an English language operating system under *Start -> Settings -> Control Panel -> Administrative Tools -> Data Sources (ODBC)*. New interfaces can be created in the category “System DSN“. It is recommended to carry out the ODBC parameterisation after the SQL database has been created. Please find below a description of how ODBC is parameterised.

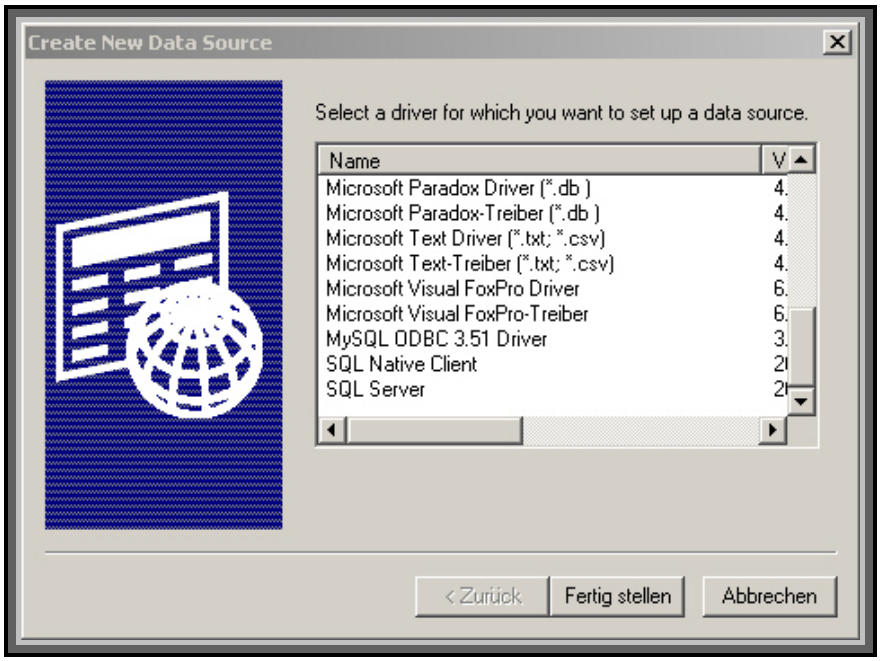
Figure 3-12 Create ODBC Interface



As can be seen in the view above several parameterised interfaces are already available. Press “Add“ to add a new one.

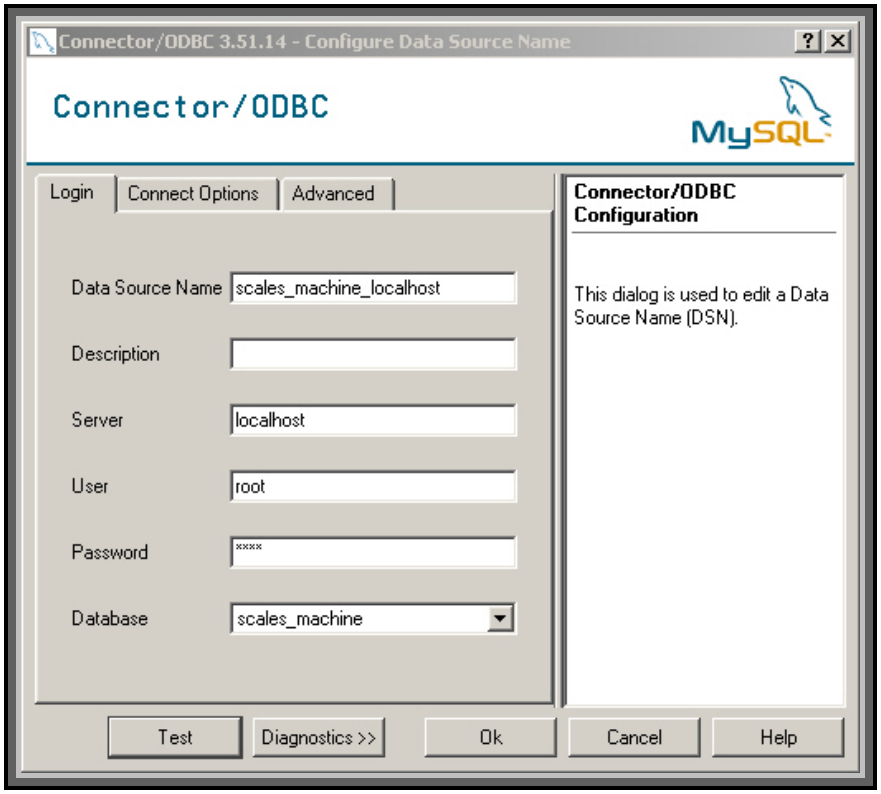
The different drivers can be selected in the following menu. In our application we are using a MySQL ODBC 3.51 Driver.

Figure 3-13 Select ODBC driver



The required parameterisations are carried out in the following window. Here you assign Data Source Name, Server, User, Password, and select the database. Now you can see the reason why we recommend creating the database first, because the driver checks straight away which databases have been made available. This is followed by checking whether the connection can be established. Please find below an example parameter assignment for a “localhost“ application.

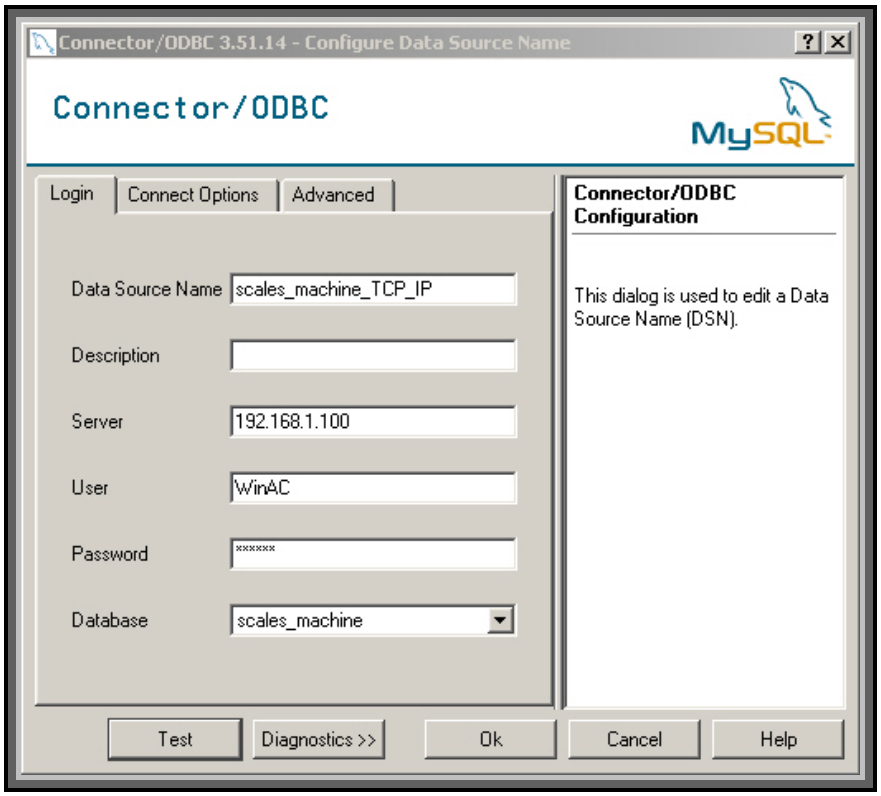
Figure 3-14 ODBC Parameterise driver (localhost)



In the next picture we parameterise the ODBC driver for a TCP/IP connection.

ATTENTION	**In order to access the SQL database via TCP/IP, it is required to previously define a user in DBMS, which in this case has been created under the name “WinAC“(see Chapter 3.2.3).**

Figure 3-15 Parameterise ODBC driver (TCP/IP)



3.4 Installation WinAC Driver on SIMATIC Engineering Computer

This documentation as well as the STEP 7 example project is required on the SIMATIC Engineering computer. The required FBs for the user's STEP 7 programme may be taken from this demo project.

ATTENTION	**The ODBC connection data in the STEP 7 project must be adapted in DB10 (DSN, User, Password). These parameters must be specified because several ODBC connections may be parameterised. When assigning a name please be aware that input is case sensitive.**

4 Installation MsSQL

4.1 Quickstart with Example Project

- Datenbase server
  - Install database system MsSQL on database server
  - Install Microsoft SQL Server Management Studio Express
  - Recover Scales Machine Schematics in MsSQL database
  - Create an SQL server logon
- Runtime-System
  - Install DLL with the DLL_Install.bat file on Runtime System
  - Parameterise SQL-Server ODBC driver on Scales Machine database
- SIMATIC Engineering Computer
  - De-archive MSSQL_BspPrj STEP 7 project on SIMATIC Engineering computer
  - Adapt ODBC connection data in DB10
  - Transfer STEP 7 project in WinAC RTX (Runtime System)

For graphic parameterisaton of the MsSQL database you can use the SQL Server Management Studio Express.

4.2 Installation of MsSQL Server and Recovery of the Schematics

4.2.1 Installation of MsSQL Database

The database system “Microsoft SQL 2005 Server Express Edition“ is installed on the database server. In order to install the database, run the file SQLEXPR_GER.exe. The programme features as illustrated in figure 4-1 are selected in the window feature selection.

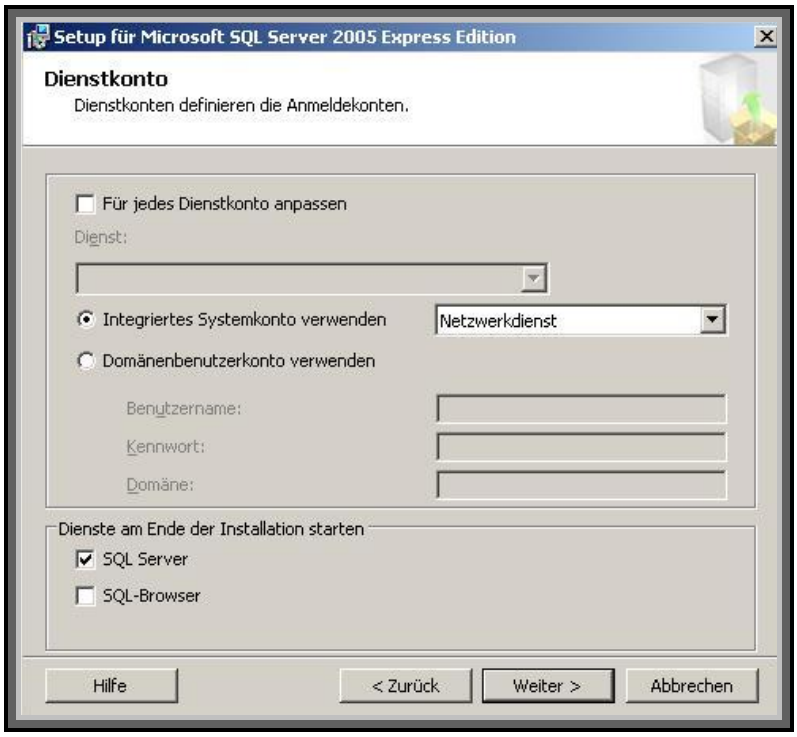
Figure 4-1 Feature selection

The instance name is maintained for the example programme.

Figure 4-2 Instance name

When setting the utility account select the network utility. For the utilities to be started you need the SQL server only for the example project.

Figure 4-3 Utility account



Select mixed as the authentication mode to ensure that both Windows and Server authentications are possible. You now need to assign a codeword for the standard user name “sa“.

Figure 4-4 Authentication mode

The following installation steps may be adopted unchanged.

After the installation you can open a console via a DOS prompt using the SQLCMD tool. This console serves to administer the database. If graphic administrations such as create and manage are required use SQL Server Management Studio Express.

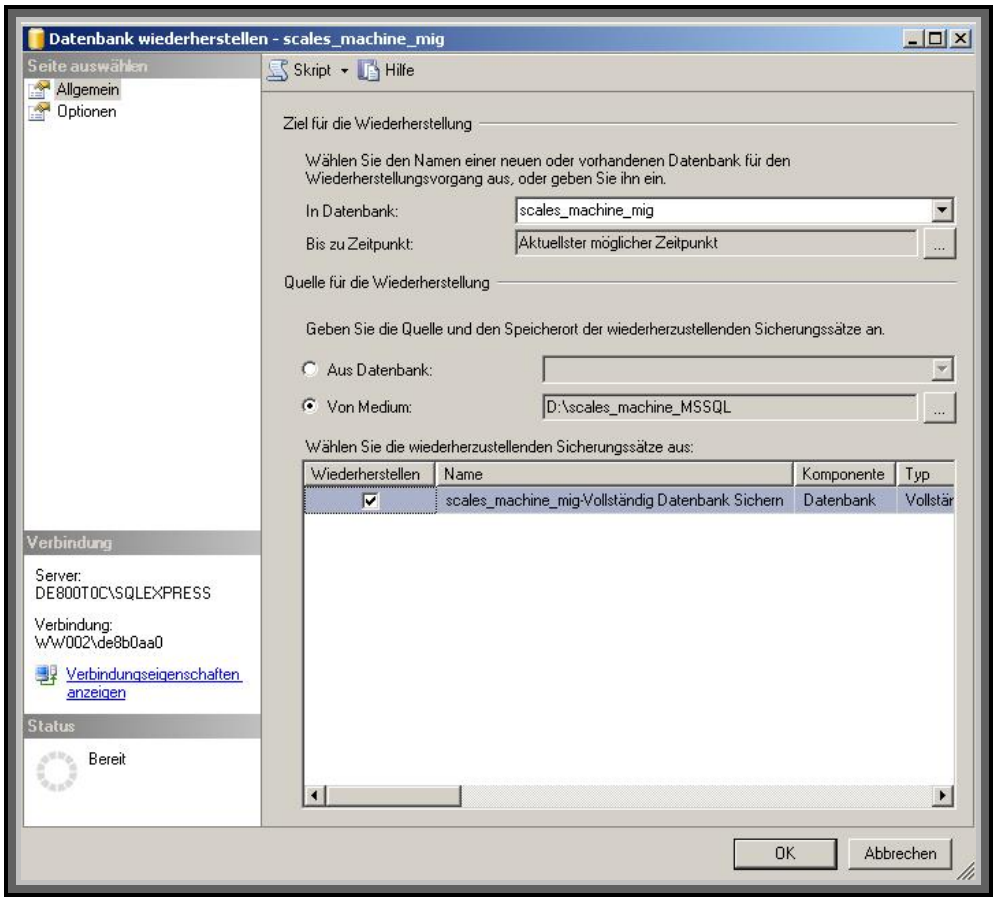
4.2.2 Installation of Microsoft SQL Server Management Studio Express

For the installation of Microsoft SQL Server Management Studio Express you will need the file SQLServer2005_SSMSEE.msi. In order to process the installation run the file and carry out the following installation steps.

4.2.3 Recovery of Example Schematic using the SQL Server Management Studio Express

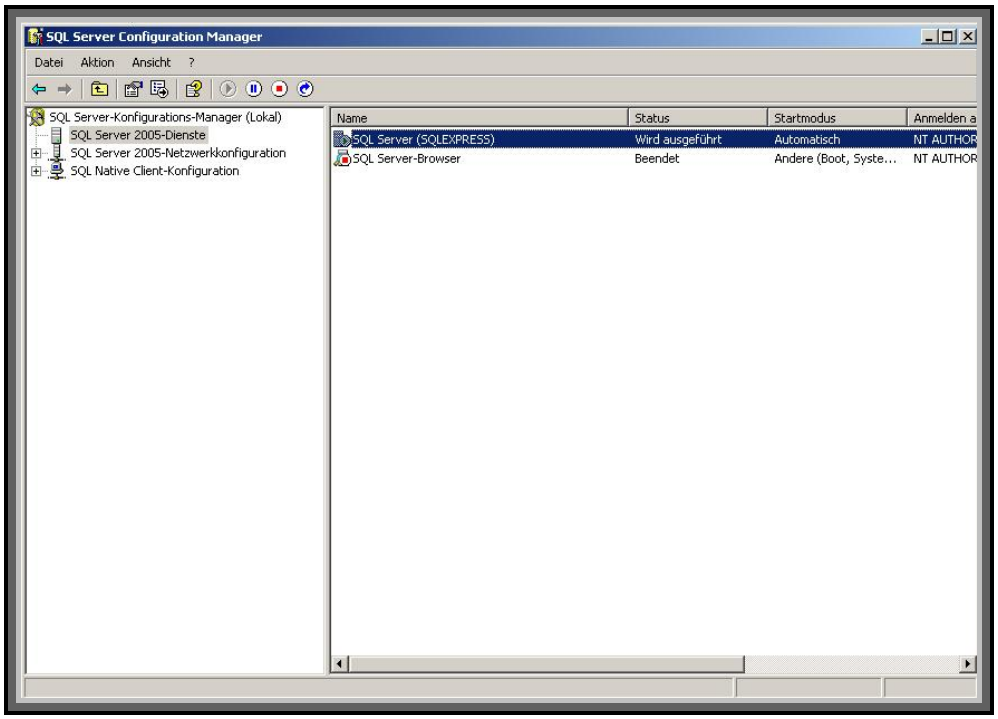
Start the SQL Server Management Studio Express to recover the saved database. Localhost is logged on Windows authentication mode, a codeword is not required. Initially you need to define a new database named “scales_machine_mig“. This is followed by the recovery of the archived database by a right mouse click on the folder of the newly defined database under Tasks -> Recover -> Database. The picture below illustrates the settings.

Figure 4-5 Database Recovery

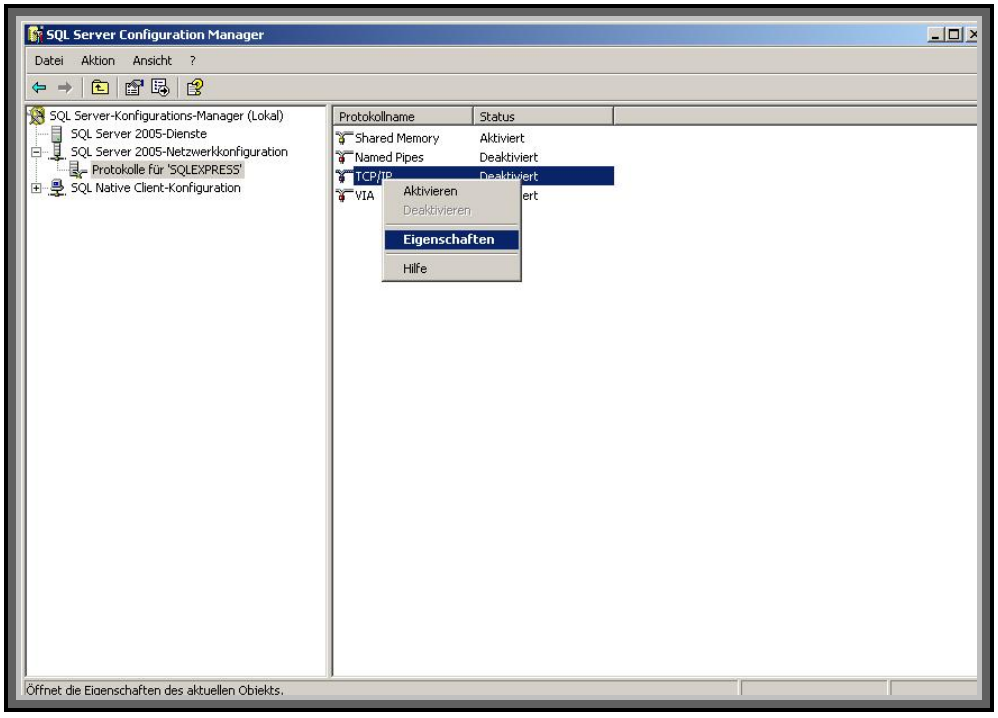


4.2.4 Important Server Settings

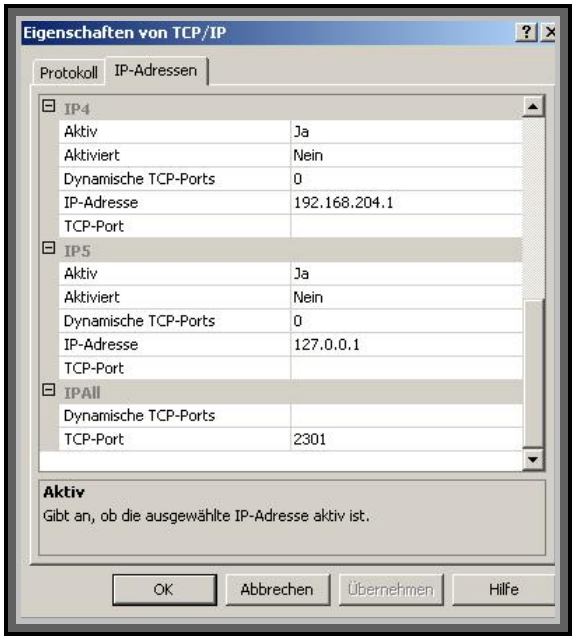
After installing MsSQL Server Express, you need to carry out settings in the SQL Server Configuration Manager. The Configuration manager is started under Start -> Programmes -> Microsoft SQL Server 2005 -> Configuration Tools.
You can view in SQL Server 2005 utilities which utilities have been started. In the case of our example programme the only utility required is SQL Server utility.



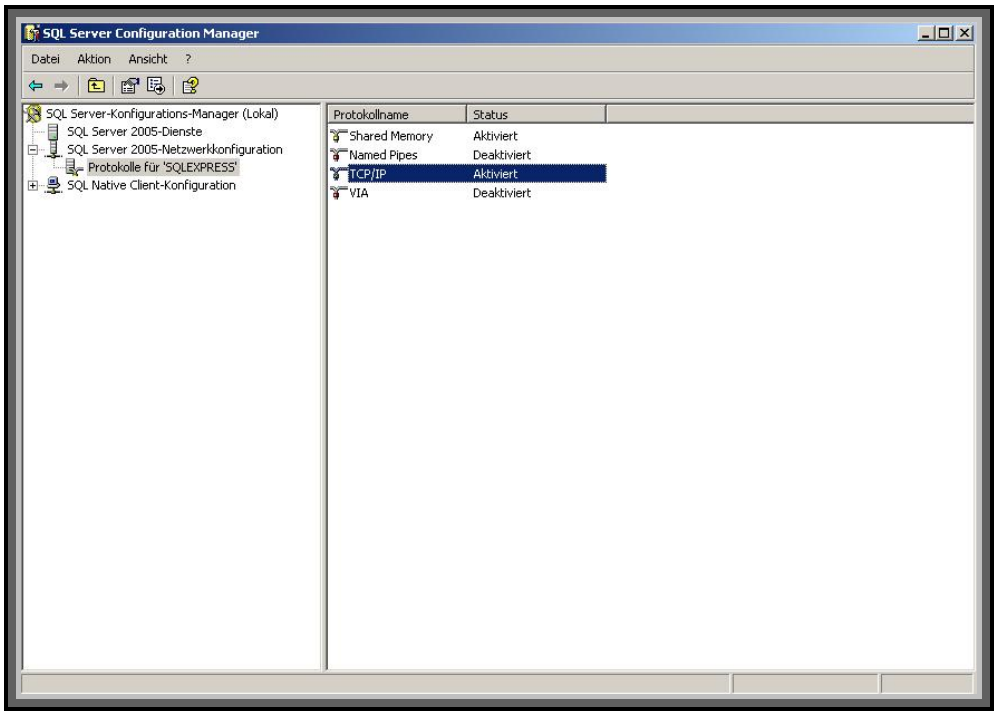
The available protocols appear under the SQL Server 2005 network configuration. As a connection should be established via TCP/IP, the appropriate utility must be parameterised. This is done by right clicking -> property.



Under tab page IP Addresses all fields for Dynamic TCP ports are deleted and port 2301 is entered into fields TCP-Ports.

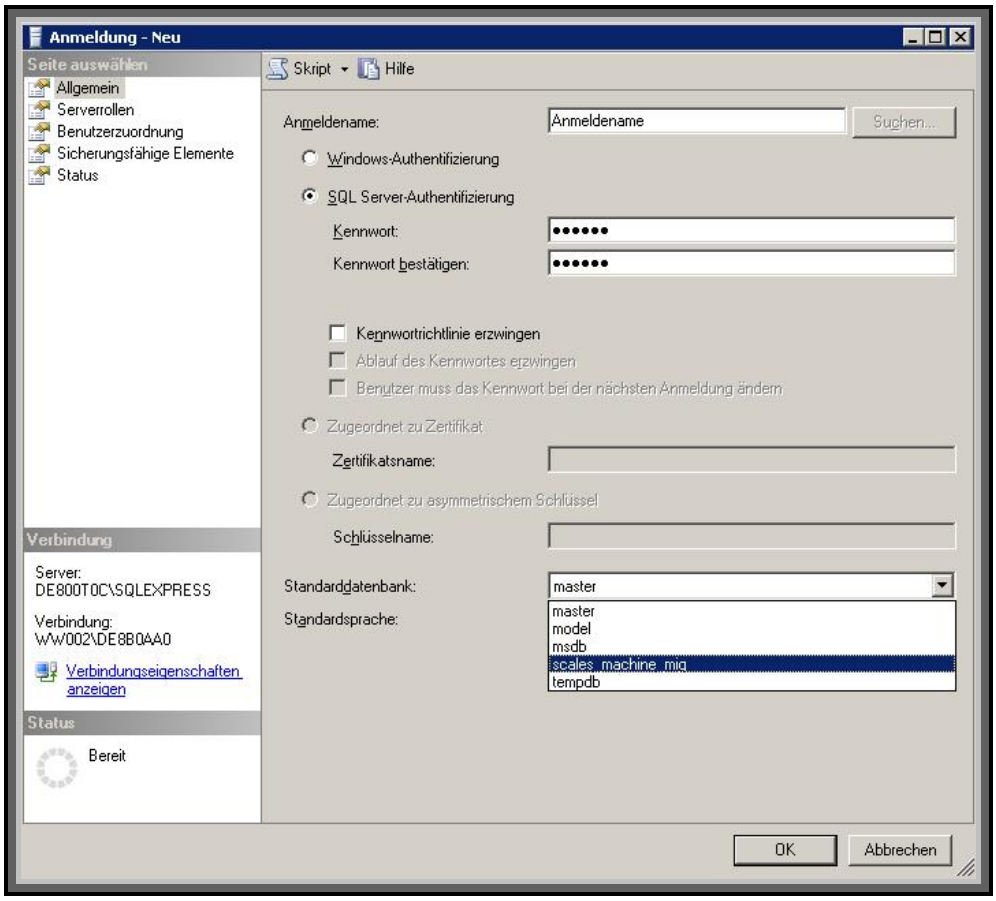


After confirming the settings, activate the protocol and re-start the SQL server.

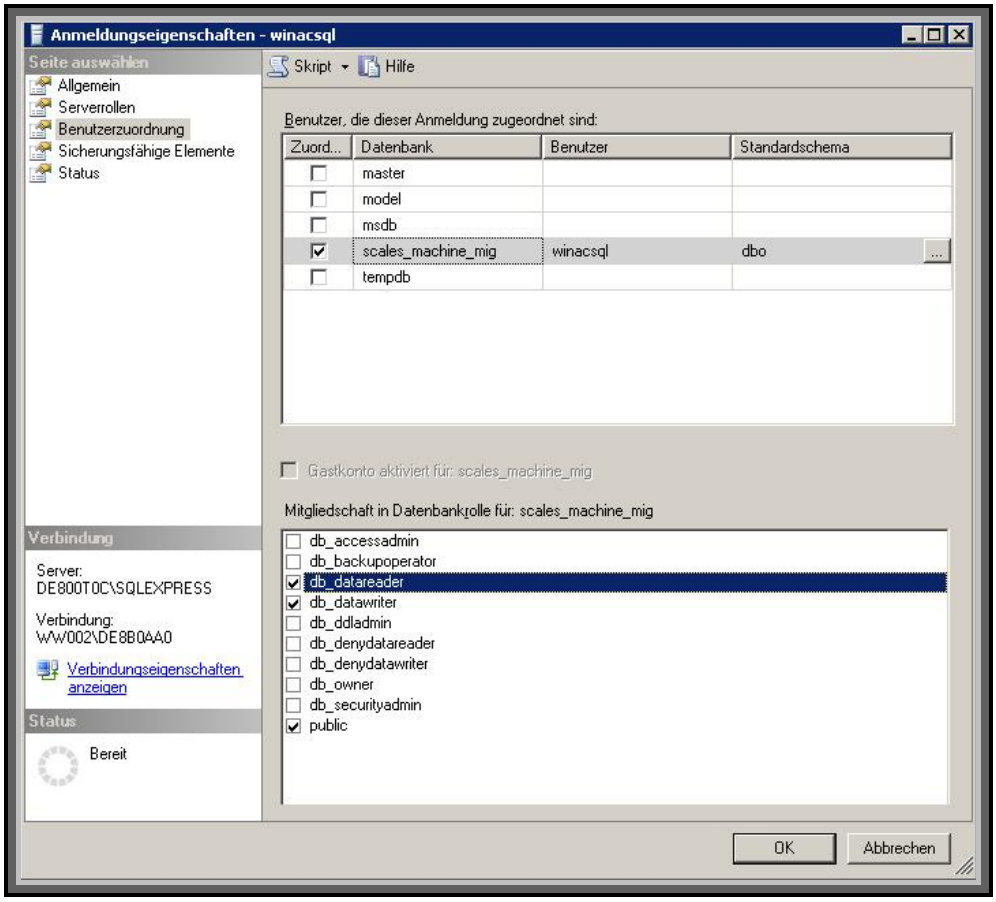


4.2.5 Create a Logon to the SQL Server

To establish a connection to the SQL server you can either go via the Windows authentication or via a SQL server authentication. For a remote connection via TCP/IP you need the SQL server authentication. For this purpose you must create a user logon.
In order to do this, open the SQL Server Management Studio Express and log on with the Windows authentication. In the left hand part of the window open the Saved folder and create a new logon by right clicking on Logon -> New Logon.
In the opened window, select the SQL server authentication. This is followed by the assignment of a logon name and code word. In our example project the recovered database is chosen as the standard database.



Under the tab page User Assignment you need to assign the appropriate authentications for the standard database in order to obtain access. This is followed by OK to create the new logon.



4.3 Installation WinAC Driver on Runtime Computer

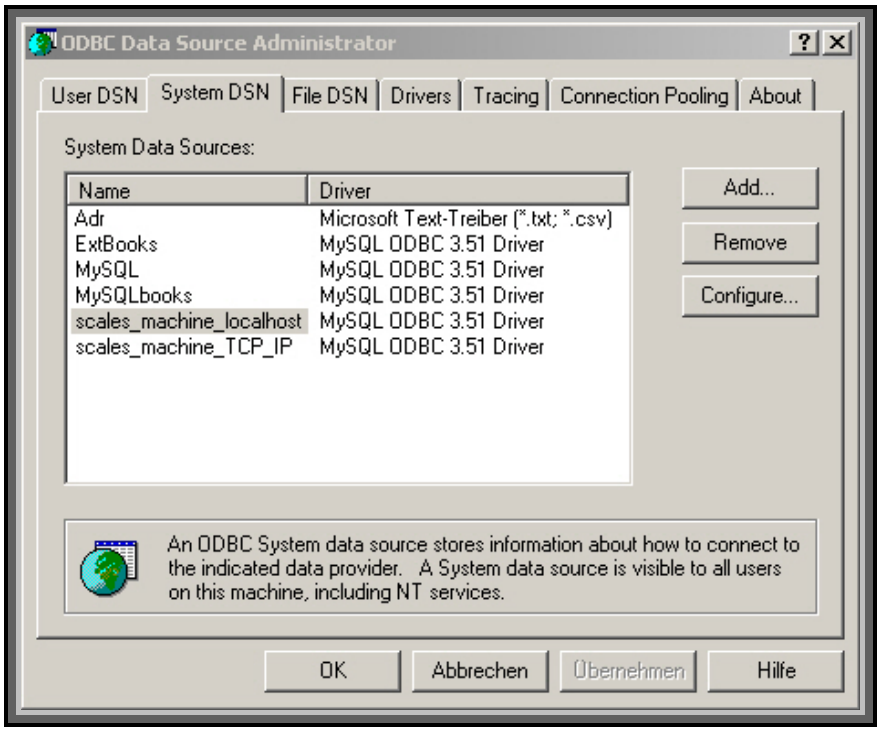
4.3.1 Installation of DLL

The installation of the WinAC driver for the SQL database connection is limited to the execution of DLL-Install.bat. During installation the DLL must be in the same directory as the bat file.
In addition the SQL Server ODBC driver must be parameterised.

4.3.2 Installation und Parameterisation of ODBC Driver

The ODBC driver is used to establish the connection to the SQL database. The ODBC Administration is located under s*ystem control\administration\data sources (ODBC)*.This is located under an English language operating system under *Start -> Settings -> Control Panel -> Administrative Tools -> Data Sources (ODBC)*. New interfaces can be created in the category “System DSN“. It is recommended to carry out the ODBC parameterisation after the SQL database has been created. Please find below the description for ODBC parameter assignment.

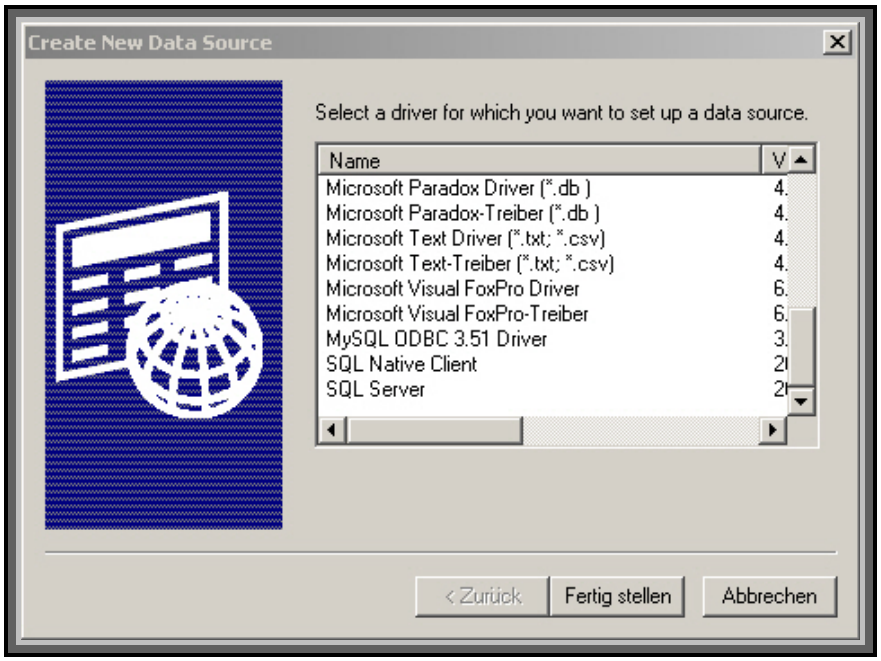
Figure 4-6 Create ODBC interface



As can be seen in the view above several parameterised interfaces are already available. Press “Add“ to add a new one.

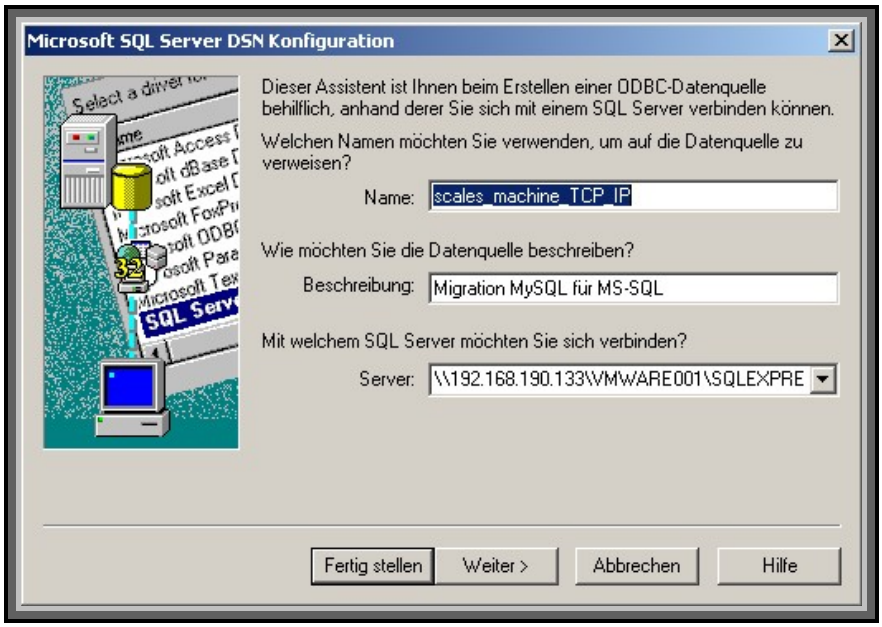
The different drivers can be selected in the following menu. In our application we are using the SQL Server.

Figure 4-7 Select ODBC Driver



The required parameterisations are carried out in the following window. Here you enter the Data Source Name, the description and the server.

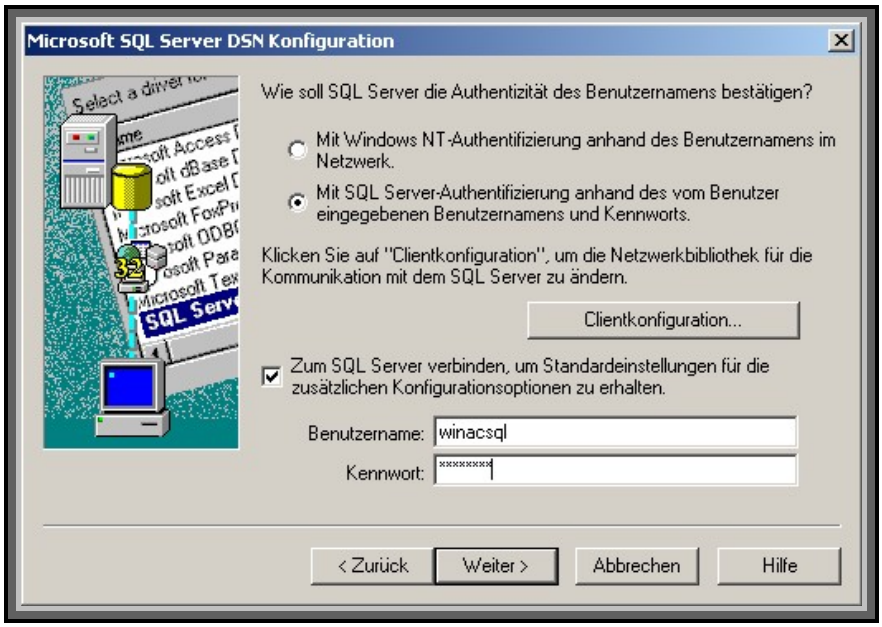
Figure 4-8 Parameterise ODBC Driver (TCP_IP)



In the next picture you select the user name/codeword logon as the authentication for the SQL-Server. This is followed by input of the user name and the codeword to allow logon. Use the logon name created in Para. 3.2.5. as the user name.

As a fixed port was entered previously in the TCP/IP protocol settings in the SQL Server Configuration Manager, you also need to specify Port 2301 in the Client Configurations.

Figure 4-9 Server logon

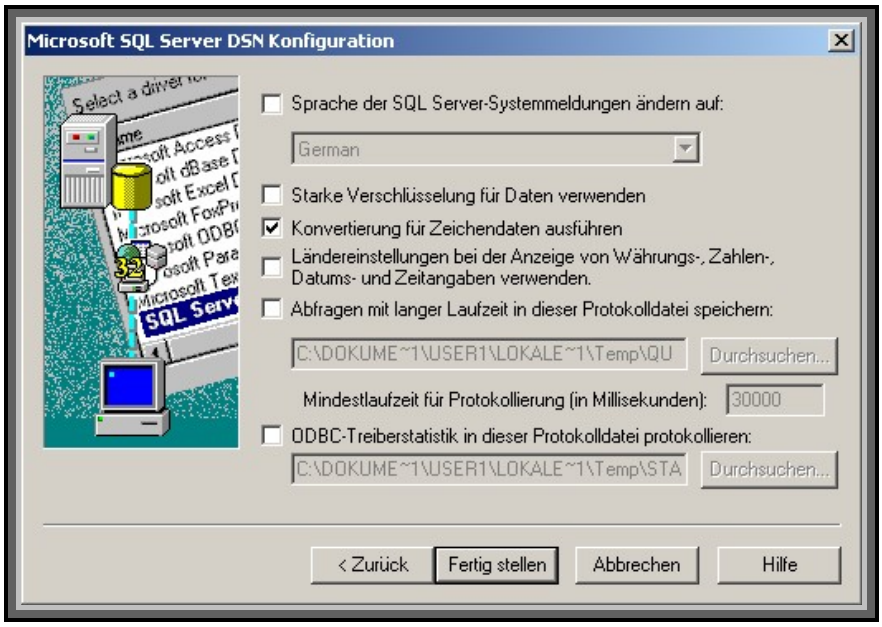


ATTENTION **In order to access the SQL database via TCP/IP, you must previously define a user in DBMS.**

In the next two steps the settings for the example project may be adopted unchanged.

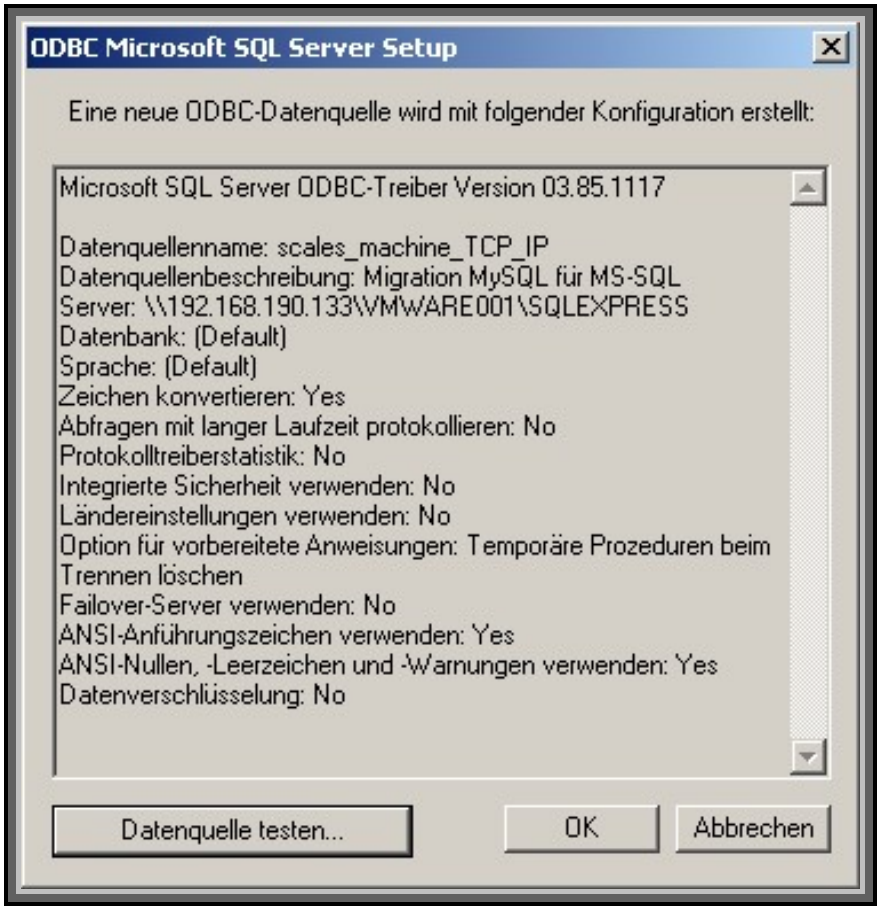
Figure 4-10 DSN Configurations

Figure 4-11 DSN Language Settings



After parameterisation has been completed a summary of the ODBC configurations is displayed where the connection to the SQL server can be tested.

Figure 4-12 DSN Summary



The test results are now displayed in a window.

Figure 4-13 DSN Test

4.4 Installation WinAC Driver on SIMATIC Engineering Computer

This documentation as well as the STEP 7 example project is required on the SIMATIC Engineering computer. The required FBs for the user's STEP 7 programme may be taken from this demo project.

ATTENTION	**The ODBC connection data in the STEP 7 project must be adapted in DB10 (DSN, User, Password). These parameters must be specified because several ODBC connections may be parameterised. When assigning a name please be aware that input is case sensitive.**

5 Functional Description

After the installation this chapter gives a rough description of how the connection to the SQL database server is established and the possible functions.

5.1 Basics

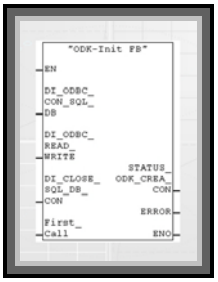
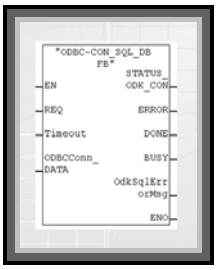
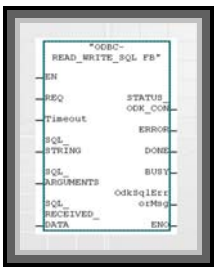
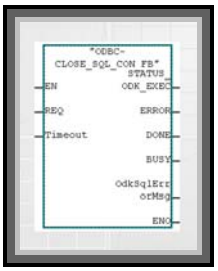
There are four function blocks available for the use of an SQL database in WinAC RTX. These function blocks serve to:

- load the DLL into the address space in WinAC RTX
- initialise required data
- establish the connection to the SQL server
- transmit SQL statements
- make available any data which have possibly been retrieved
- close the database connection.

ATTENTION	**In order to avoid any buffer overflow always close the SQL database with the appropriate function block. This also applies in the event of an error.**

The listed functions are allocated to the function blocks as illustrated in the table below:

Table 5-1 Overview of FBs

Screen	Name	Function
	SQL_INIT	• load DLL into address space of WinAC RTX • initialise required data
	SQL_CON	• establish connection to SQL server
	SQL_EXEC	• transmit SQL statements • make available any possibly retrieved data
	SQL_DISCON	• close database connection

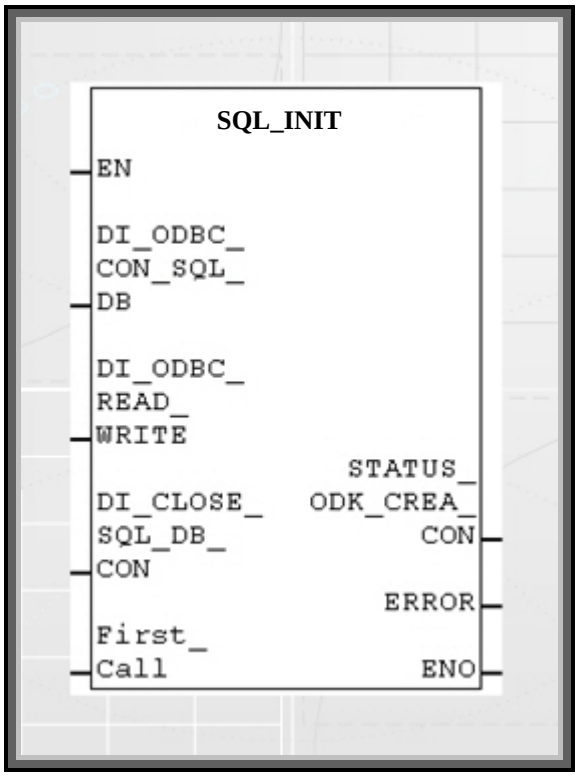
6 Detailed Description of FBs

6.1 ODK Initialisation Block

The ODK initialisation block is the first block which must be processed in the step sequence. It loads the DLL file to the address space in WinAC RTX and distributes the returned Handle to the other ODBC-FBs. It also initialises all error memories and retentive data in the different instance data blocks (DI) of the ODBC-FBs, including its own DI. The initialisation block and the FB, which establishes the connection to the SQL server, have been separated on purpose. The advantage of this separation is an improved evaluation of any errors which may occur in the step sequence. It may happen that the DLL has already been loaded into the address area of WinAC, but the SQL server link has been interrupted and has to be re-established. The DLL does not have to be re-loaded into the address space after the Handle has been successfully returned to CREA_COM; the only thing which needs to be re-established is the connection to the SQL server.

To ensure that SQL_INIT is only processed once at the beginning, this block should be called from OB100 (responsible for the warm re-start in WinAC).

Figure 6-1 ODK Initialisation



Prior to examining the syntactical part of the block, let us first look at the interfaces.

Input parameter SQL_INIT

- Transfer of instance data block number

  The DI numbers of the three other function blocks which belong to the ODBC blocks, must be allocated to the following DI inputs so that the SQL_INIT recognises which DIs must be initialised.

- First Call

  The First_Call determines whether a Handle for communication build up must be generated or whether only the data blocks must be re-initialised. Output parameter SQL_INIT

- Status of CREA_COM

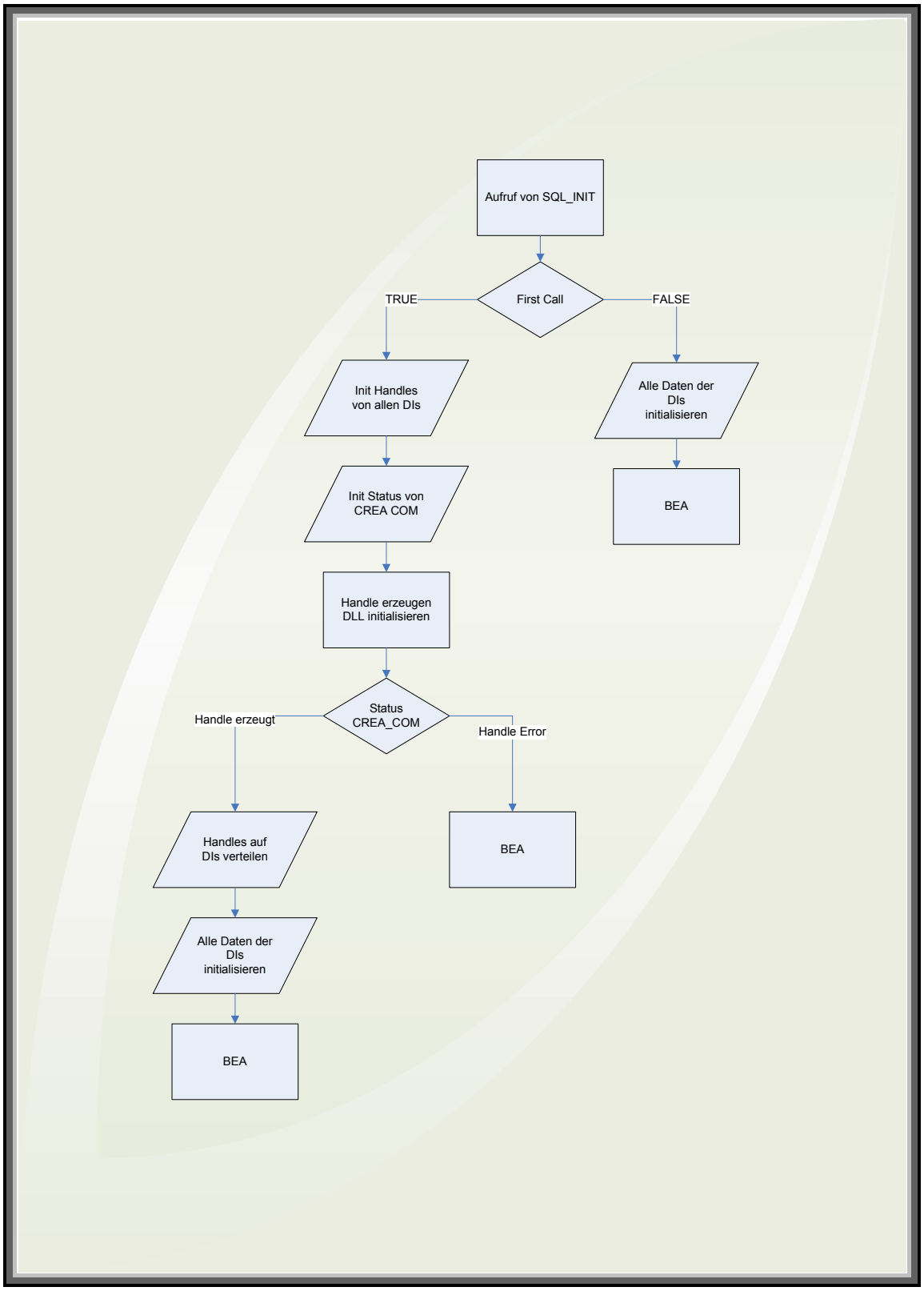
  The output STATUS_ODK_CREA_COM supplies the handle number or, if the build up was unsuccessful, it returns an appropriate error code. The error code is listed in the WinAC ODK User Manual and can be looked up there. The data format of the output variable must comply with WORD.

- ERROR

  The ERROR-Bit specifies whether the SQL_INIT was processed successfully or whether it was interrupted with error. If TRUE is output there has been an error.

The following block diagram illustrates the syntactical sequence.

Figure 6-2 Flowchart of SQL_INIT



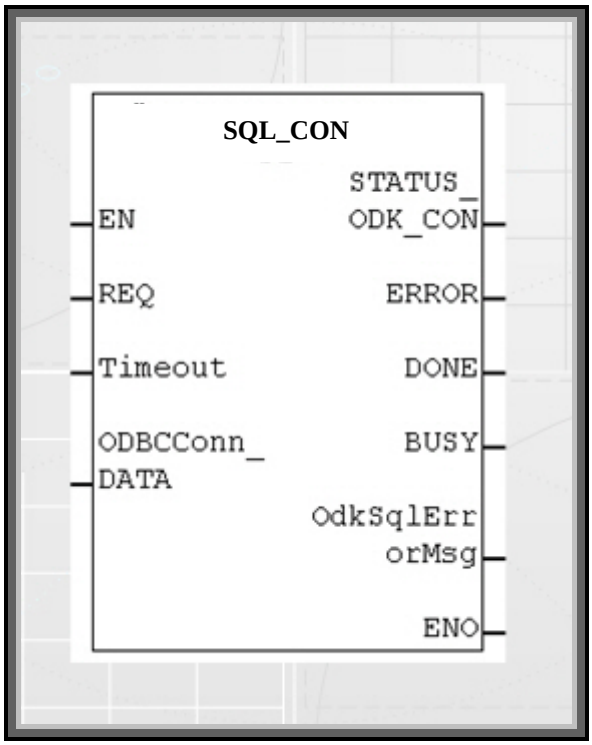
6.2 ODBC Communication Build-up to the SQL Datenbase

The ODBC-Con_SQL_DB function block is responsible for the build-up of the ODBC interface. It is called to establish a connection to a MySQL-DBMS. Prior to the build-up to DBMS it is called only once in the step sequence. After the connection has been successfully established, the block is not processed again until the connection to DBMS has either been interrupted or broken. In order to establish a connection to a DBMS, system-DSN, user name and password must be transferred. This happens via a DB, which has saved this data in string format. The block is processed asynchronously to the running PLC cycle, i.e. a thread is triggered which must have established the connection following a specified time (Timeout). Due to the fact that there may be considerable time fluctuations with a connection which is established via a TCP/IP connection, but the PLC cycle must not be interrupted for longer periods of time, it is recommended to process the DQL database connection call synchronously to the PLC cycle.

Note

An SQL database connection is supported. If a new connection is established there will be an error message.

Figure 6-3 ODBC Connection Block



Prior to examining the syntactical part of the block, let us first look at the interfaces.

Attention **The EN input and the ENO output are not parameterised.**

Input Parameters SQL_CON

- REQ

  Input REQ is the On/Off switch on the FB. If there is a logical One at the input, the block is processed.

- Timeout

  An integer value is set at this input, which indicates the maximum asynchronous processing time of the block in seconds.

- ODBC Parameter Transfer

  A pointer to a DB is transferred at ODBCConnDATA. This pointer contains the connection parameters for the ODBC interface.

Figure 6-4 ODBC-Connection Data in a DB

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	OdbcSystemDSN	STRING[254]	'weighting_machine'
+256.0	UserName	STRING[254]	'WinAC'
+512.0	PassWord	STRING[254]	'diplom'
=768.0		END_STRUCT	

As can be seen in Figure 5-4, the parameters are transferred in string format whereby each individual parameter is written in a string of its own which is 254 characters long. The naming of the individual parameters is therefore limited to a length of 254 characters (max. character length in STEP 7). The actual length of the individual string variable amounts to two more bytes because the information for the maximum string variable length and the actual string variable length are stored in a character string (first and second byte). Please note that for the parameterisation of the pointer the entire length of the string is specified because otherwise characters are chopped off during transmission. As a reference value you can use the specified address in the DB under the last string.

Output Parameters SQL_CON

- Status of EXEC_COM

  In STATUS_ODK_CON error messages are returned by EXEC_COM. In the event of successful calling and processing, a Zero is displayed by the output; in the event of an error an appropriate error code is returned. The error code is listed in the WinAC ODK User Manual and can be looked up there. The data format of the output variable must comply with WORD.

- ERROR

  The ERROR-Bit indicates whether the SQL_CON was processed successfully or whether it was interrupted with error. If a One is output, there has been an error during processing.

- DONE

  As soon as all cycles in SQL_CON have been completed, the DONE output is set to TRUE. This also happens when it is interrupted by an error. Due to the fact that the last cycle was completed in a defined way, DONE is set to TRUE nevertheless with the additional information that ERROR equals TRUE.

- BUSY

  As several block calls are necessary, the block must indicate whether it is still in the processing phase, i.e. the BUSY-Bit is set to TRUE during the entire processing of the block and is only reset to FALSE until the block has been definitely completed.

- Error buffers for ODBC error messages

  The output OdkSqlErrorMsg corresponds to a structure which may contain several error codes from the DLL. In order to forward this data to a DB, a User Defined data type (UDT) is required. This UDT is structured as illustrated in Figure 5-5:

Figure 6-5 UDT Structure of Error-Struct

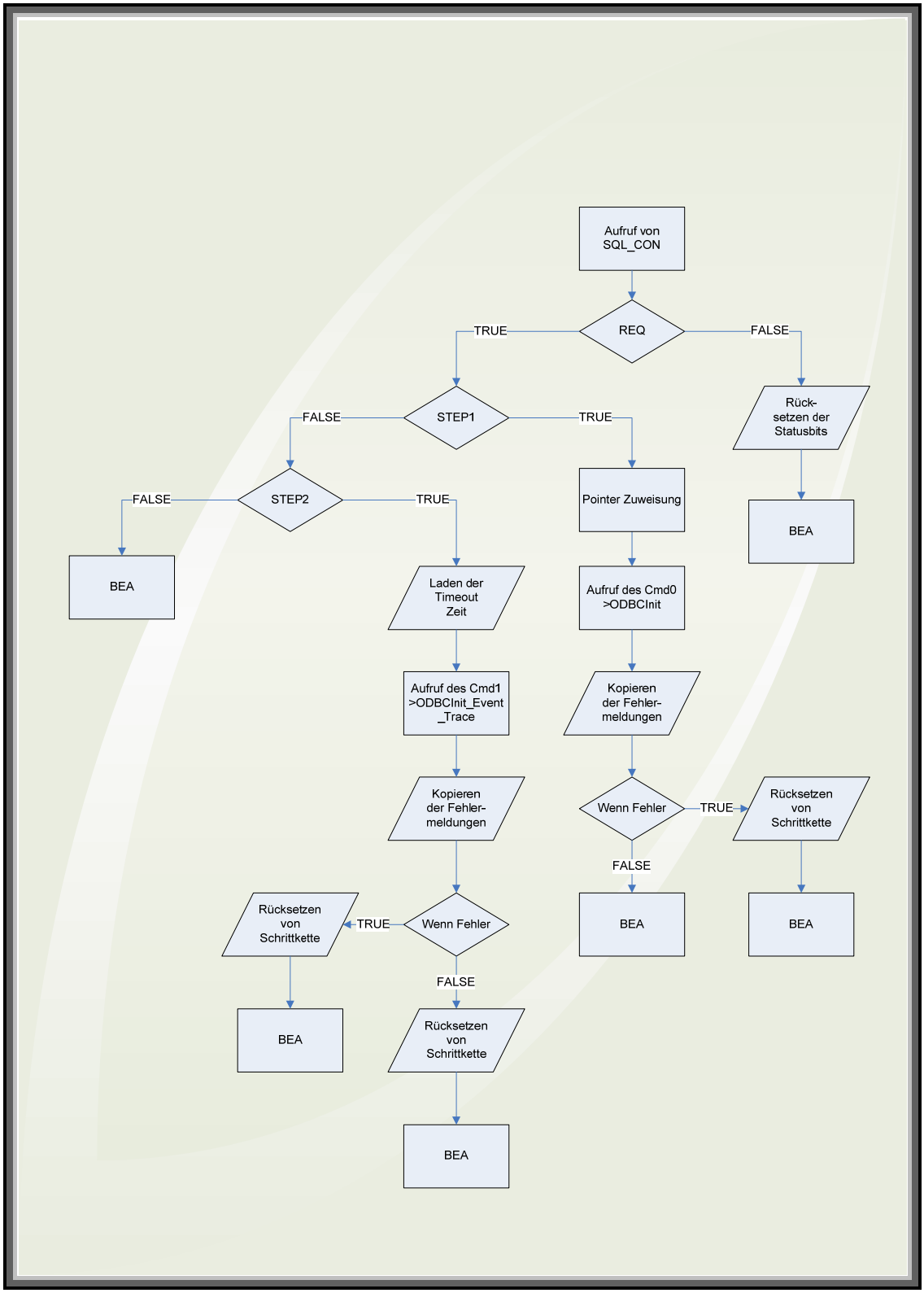
Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	OdkFctRETURN	DWORD	DW#16#0
+4.0	OdbcFctnum	DWORD	DW#16#0
+8.0	OdbcSqlDefineErrorCode	DWORD	DW#16#0
+12.0	OdbcUserDefineErrorCode	DWORD	DW#16#0
+16.0	ODBCSqlDefineErrorSt1	STRING[6]	''
+24.0	ODBCSqlDefineErrorSt2	STRING[6]	''
+32.0	ODBCSqlDefineErrorSt3	STRING[6]	''
+40.0	ODBCSqlDefineErrorSt4	STRING[6]	''
+48.0	ODBCSqlDefineErrorSt5	STRING[6]	''
+56.0	ODBCSqlDefineErrorSt6	STRING[6]	''
+64.0	ODBCSqlDefineErrorSt7	STRING[6]	''
+72.0	ODBCSqlDefineErrorSt8	STRING[6]	''
+80.0	ODBCSqlDefineErrorSt9	STRING[6]	''
+88.0	ODBCSqlDefineErrorSt10	STRING[6]	''
=96.0		END_STRUCT	

Now if this UDT was created in a DB, SQL-CON is parameterised with the appropriate UTD. This is followed by the transmission of the error codes to the DB.

For the individual error numbers and error statements please refer to tables 7-3 to 7-6 in the appendix where you will find the necessary explanations.

The following block diagram illustrates the syntactical sequence.

Figure 6-6 Flowchart of SQL_CON



6.3 ODBC Read and Write Block for SQL Database

The central block SQL_EXEC is responsible for all functions which are required for sending SQL manipulation commands and for receiving possibly requested data. After successful connection to the SQL database, SQL_EXEC is executed in order to process the database. This block may be called as often as required to carry out SELECT-, UPDATE-, INSERT- or DELETE statements. A statement is processed asynchronously to avoid exceeding the cycle time of WinAC. Several cycles are processed to execute and complete SQL_EXEC. In order to define a maximum waiting time for the processing of the asynchronous Threads, a Timeout at the block is defined. After the specified time has elapsed, the block is interrupted and an error message is sent.

SQL statements consist of two parts; the SQL string and the SQL argument. The actual commands are packed in the SQL-String, such as.

```
SELECT columntitle1, columntitle2
  FROM tabletitle.
```

As it should be possible to change or read certain areas from the database, the SQL string must be variable to enable the user to select from different criteria. For instance you may only look for data which were saved within a certain period of time. The first query would search between the months of May and July and the second between October and December. This is possible by means of the arguments which are input separately from the SQL string and are combined with the SQL string in SQL_EXEC. As an additional liberty it is also possible to use different data formats for the arguments. The following are possible: -

SMALLINT-, INTEGER-, REAL- and CHAR data types.

Example: a statement may look like this:

```
SELECT year, test
  FROM %s
 WHERE titleid < %d
```

"%s" and *"%d"* are placeholders for the coming arguments which may be:

```
UDT STRING(254)  => titles
UDT SMALLINT          => 5
```

The complete statement for the database may be interpreted as follows:

```
SELECT year, test
  FROM titles
 WHERE titleid < 5
```

The arguments are inserted in the respective placeholders whereby the length of the individual arguments is limited to the appropriate data type. A special case is STRING where up to 254 characters per argument may be used. Please note that the arguments must be in the same order as the placeholders. You will get an error message when the data type does not correspond to the placeholder. The following table illustrates corresponding placeholders and data types.

Table 6-1 corresponding data types and placeholders

Data types	Placeholders	Defined Data type number
INTEGER	%d	1
SMALLINT	%d	2
REAL	%f	4
CHAR (STRING)	%s	5

The amount of placeholders must also match the arguments.

In order to achieve conformity is necessary to use four pre-defined UDTs which correspond to the above mentioned data types. As the first piece of information the UDTs contain the data type as a defined number whereby “1“ corresponds to INTEGER, “2“ to SMALLINT, “4“ to REAL and “5“ to CHAR (STRING). “3“ is reserved for type DOUBLE (64Bit) but as this is not supported by WinAC it is converted to REAL (32Bit). The second piece of information is the value of the argument which is stored in the appropriate data format. The following picture illustrates the internally defined data type UDT-STRING.

Figure 6-7 UDT4 String

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	typCh	INT	5
+2.0	data	STRING[254]	''
=258.0		END_STRUCT	

When a SELECT statement is executed returned data are expected by the block. It is necessary to make a DB available which has to be of a suitable data type. The data types correspond to the UDTs of the arguments and are therefore used for saving the data. The UDTs in the DB must be available in the same order as the data are supplied by the block. I.e. the block buffers the required data in a certain order, which must be known. If the required data do not match the data which has been made available, there will be an error message. Error messages are also output in the event of too few or too many provided UDTs. An SQL database supports several data types. The following SQL data types are supported by the application:

- CHAR
- NUMERIC
- DECIMAL
- VARCHAR
- INTEGER
- SMALLINT

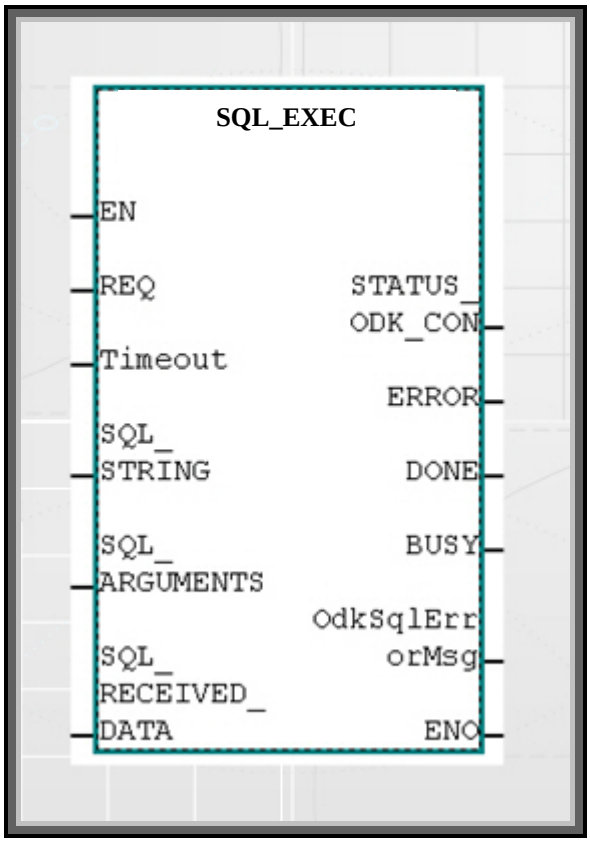
- FLOAT
- REAL
- DOUBLE

The data types DOUBLE and REAL relating to SQL consist of 64Bit. These are not supported by STEP 7 and are converted to STEP 7-REAL(32Bit). Due to the conversion of a 64Bit number to a 32Bit number the value loses accuracy.

Attention **When using data type DOUBLE or REAL on SQL level, the values are converted to STEP 7-REAL in WinAC. The consequence is loss of accuracy of the converted value.**

Note The data type FLOAT in SQL level corresponds to data type REAL on der STEP 7 level.

Figure 6-8 ODBC-READ and WRITE Block



Prior to examining the syntactical part of the block, let us first look at the input and output parameters again:

Attention **The EN input and the ENO output are not parameterised.**

Input Parameter SQL_EXEC

- REQ

  Input REQ is the On/Off switch on the FB. If there is a logical One at the input, the block is processed.

- Timeout

  An integer value is set at this input, which indicates the maximum asynchronous processing time of the block in seconds.

- SQL_STRING

  A pointer to a DB is transferred to a DB. This contains four Strings, in which the SQL String is written. The maximum length of an SQL String is 1016 characters. Please note that the correct length is transferred to the pointer. As a reference value you can use the address which is listed last in the address column of the DB.

Figure 6-9 DB for SQL String

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	SQL_String1	STRING[254]	'SELECT year, test FROM titles WHERE titleid < %s%d'
+256.0	SQL_String2	STRING[254]	''
+512.0	SQL_String3	STRING[254]	''
+768.0	SQL_String4	STRING[254]	''
=1024.0		END_STRUCT	

- SQL_ARGUMENTS

  The arguments are transferred to a pointer which points to a DB. Please note that the arguments must be arranged in the same order as the placeholders. The amount of placeholders must also match the arguments. An unlimited number of arguments may be entered. It is also allowed to have various data types in a string. It is important to note that the correct length is transferred to the pointer. As a reference value you can use the address which is listed last in the address column of the DB.

ATTENTION **The maximum amount which may be used on arguments is limited due to the maximum data storage in one DB (65 Kbytes) and by a maximum amount of 1000 arguments.**

Figure 6-10 DB for SQL Arguments

Adresse	Name	Typ
0.0		STRUCT
+0.0	ARG_STR_1	"Data Type Char"
+258.0	ARG_INT_1	"Data Type Integer"
=264.0		END_STRUCT

- SQL_RECEIVED_DATA

  A pointer is transferred which points to a DB which contains a certain amount of UDTs, where read data can be stored. Therefore the SQL_RECEIVE_DATA is in its actual sense an output because read data are being made available.

The UDTs in the DB must exist in the same order as data is supplied by the block. The amount of supplied data must also match the provided UDTs. It is important to note that the correct length is transferred to the pointer. As a reference value you can use the address which is listed last in the address column of the DB. The following picture illustrates a Receive which is provided for four values. The UDTs have been arranged in the following order: INTEGER; CHAR; INTEGER; CHAR. The data is expected in exactly that order.

Attention **The maximum amount of values which may be returned in one query is limited due to the maximum data storage in one DB (65 Kbytes) and by a maximum amount of 1000 values.**

Figure 6-11 DB for SQL-Receive

Adresse	Name	Typ
0.0		STRUCT
+0.0	SQL_field2	"Data Type Integer"
+6.0	SQL_field4	"Data Type Char"
+264.0	SQL_field21	"Data Type Integer"
+270.0	SQL_field41	"Data Type Char"
=528.0		END_STRUCT

Output Parameter SQL_EXEC

- Status of EXEC_COM

  Error messages from EXEC_COM are returned in STATUS_ODK_CON. In the event of successful calling and processing, a Zero is displayed by the output; in the event of an error an appropriate error code is returned. The error code is listed in the WinAC ODK User Manual and can be looked up there. The data format of the output variable must comply with WORD.

- ERROR

  The ERROR-Bit indicates whether the EXEC_COM was processed successfully or whether it was interrupted with error. If a One is output, there has been an error during processing.

- DONE

  As soon as all cycles have been completed by EXEC_COM, the DONE output is set to TRUE. This also happens when it is interrupted by an error. Due to the fact that the last cycle was completed in a defined way, DONE is set to TRUE nevertheless with the additional information that ERROR equals TRUE.

- BUSY

  As several block calls are necessary, the block must indicate whether it is still in the processing phase, i.e. the BUSY-Bit is set to TRUE during the entire processing of the block and is only reset to FALSE until the block has been definitely completed. It is also required to ensure that only one SQL statement at a time is being processed. It is essential to wait for the BUSY-Bit, until the next SQL-Statement can be sent off.

- Error Buffer for ODBC Error Messages

The output OdkSqlErrorMsg corresponds to a structure which may contain several error codes from the DLL. In order to forward this data to a DB, a User Defined data type (UDT) is required. This UDT is structured as illustrated in Figure 5-12:

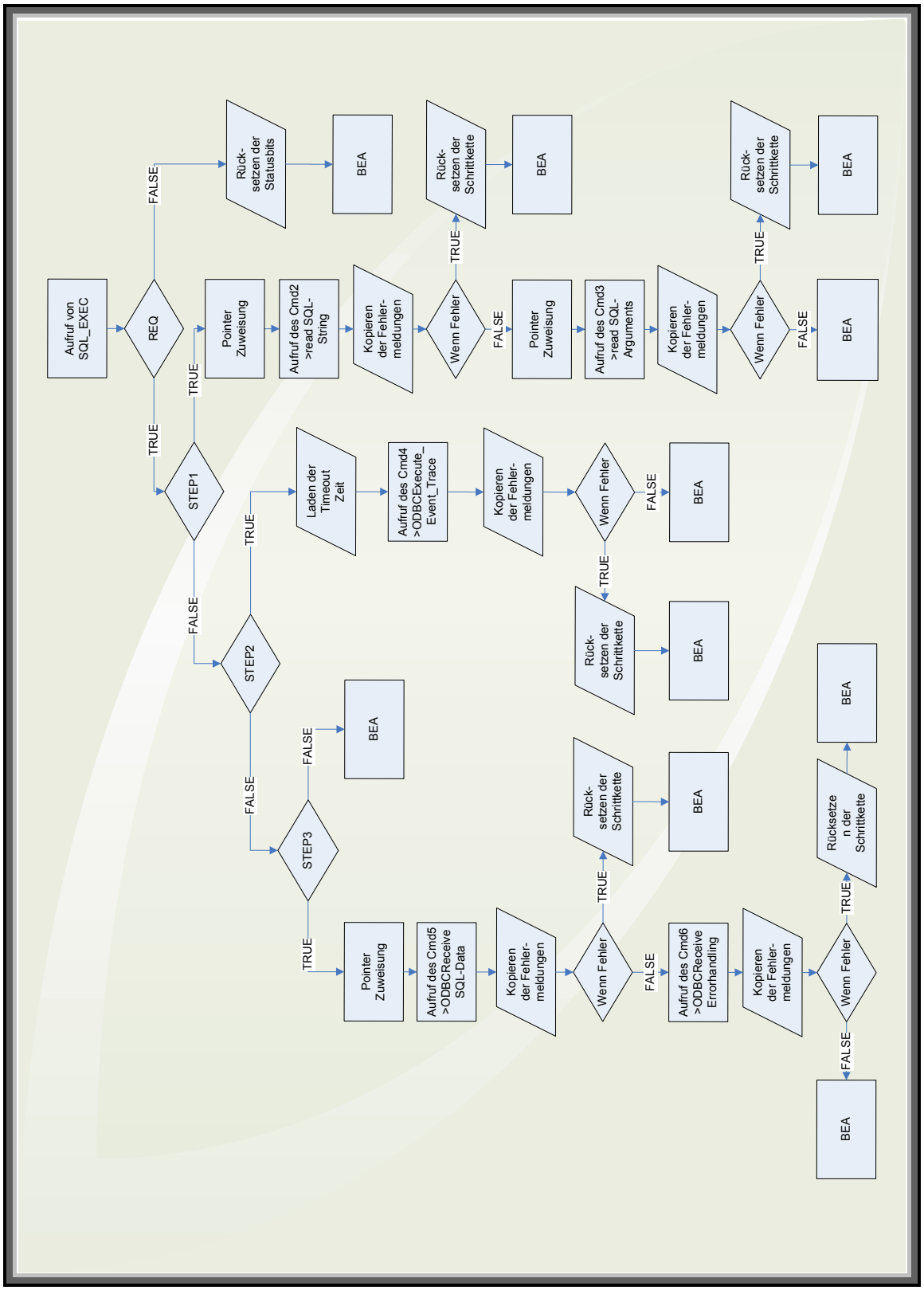
Figure 6-12 UDT Structure of Error-Struct

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	OdkFctRETURN	DWORD	DW#16#0
+4.0	OdbcFctnum	DWORD	DW#16#0
+8.0	OdbcSqlDefineErrorCode	DWORD	DW#16#0
+12.0	OdbcUserDefineErrorCode	DWORD	DW#16#0
+16.0	ODBCSqlDefineErrorSt1	STRING[6]	''
+24.0	ODBCSqlDefineErrorSt2	STRING[6]	''
+32.0	ODBCSqlDefineErrorSt3	STRING[6]	''
+40.0	ODBCSqlDefineErrorSt4	STRING[6]	''
+48.0	ODBCSqlDefineErrorSt5	STRING[6]	''
+56.0	ODBCSqlDefineErrorSt6	STRING[6]	''
+64.0	ODBCSqlDefineErrorSt7	STRING[6]	''
+72.0	ODBCSqlDefineErrorSt8	STRING[6]	''
+80.0	ODBCSqlDefineErrorSt9	STRING[6]	''
+88.0	ODBCSqlDefineErrorSt10	STRING[6]	''
=96.0		END_STRUCT	

Now if this UDT was created in a DB, SQL_EXEC is parameterised with the appropriate UTD. This is followed by the transmission of the error codes to the DB. For the individual error numbers and error statements please refer to tables 7-3 to 7-6 in the appendix where you will find the necessary explanations.

The following block diagram illustrates the syntactical sequence.

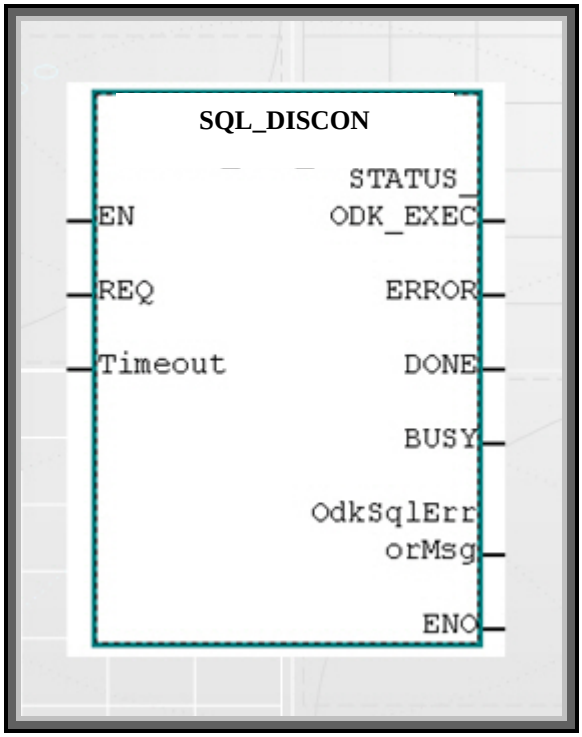
Figure 6-13 Flow Chart of ODBC-READ_WRITE



6.4 ODBC Block for Closing the Communication

The last block in the step sequence is SQL_DISCON. It is responsible for the secure closing of the database. When a connection to an SQL database is no longer required, it must be closed again with this block. If the connection is not closed again, the established connections to the SQL server are not terminated. This may lead to errors such as *Too Many Client Conenctions Active*. It can also result in a buffer overflow, because several allocated storage areas are required for the connections, which are not properly closed without processing. SQL_DISCON should be called in any case.

Figure 6-14 ODBC-CLOSE Block



Prior to examining the syntactical part of the block, let us first look at the input and output parameters again:

Attention **The EN Input and the ENO Output are not parameterised.**

Input SQL_DISCON

- REQ

  Input REQ is the On/Off switch on the FB. If there is a logical One at the input, the block is processed.

- Timeout

  An integer value is set at this input, which indicates the maximum asynchronous processing time of the block in seconds.

  Output Parameter SQL_DISCON

- Status des EXEC_COM

  In STATUS_ODK_CON error messages are returned by EXEC_COM. In the event of successful calling and processing, a Zero is displayed by the output; in the event of an error an appropriate error code is returned. The error code is listed in the WinAC ODK User Manual and can be looked up there. The data format of the output variable must comply with WORD.

- ERROR

  The ERROR-Bit indicates whether the SQL_DISCON was processed successfully or whether it was interrupted with error. If a One is output, there has been an error during processing.

- DONE

  As soon as all cycles in SQL_DISCON have been completed, the DONE output is set to TRUE. This also happens when it is interrupted by an error. Due to the fact that the last cycle was completed in a defined way, DONE is set to TRUE nevertheless with the additional information that ERROR equals TRUE.

- BUSY

  As several block calls are necessary, the block must indicate whether it is still in the processing phase, i.e. the BUSY-Bit is set to TRUE during the entire processing of the block and is only reset to FALSE until the block has been definitely completed.

- Error Buffer for ODBC Error Messages

  The output OdkSqlErrorMsg corresponds to a structure which may contain several error codes from the DLL. In order to forward this data to a DB, a User Defined data type (UDT) is required. This UDT is structured as illustrated in Figure 5-12:

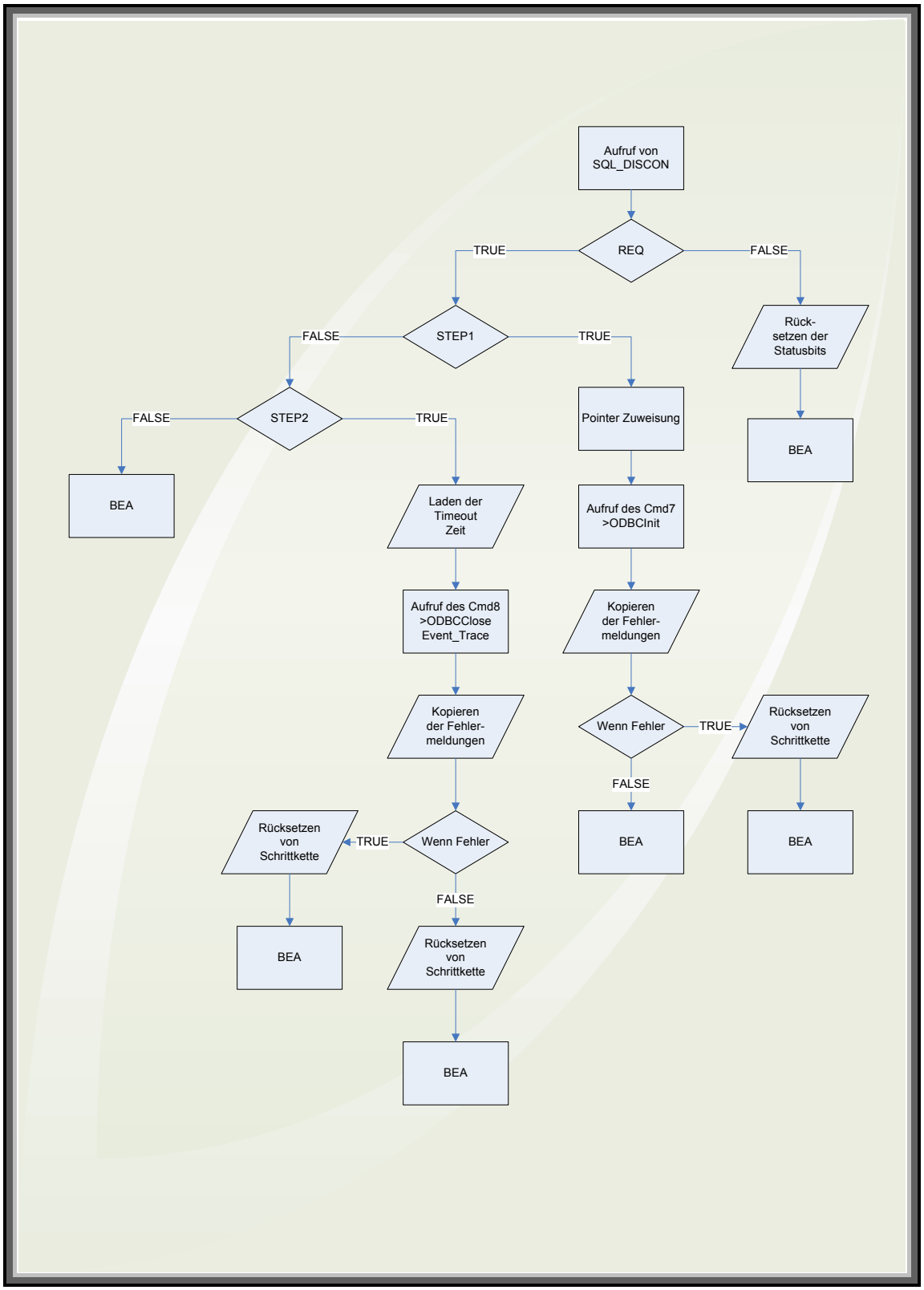
Figure 6-15 UDT Structure of Error-Struct

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	OdkFctRETURN	DWORD	DW#16#0
+4.0	OdbcFctnum	DWORD	DW#16#0
+8.0	OdbcSqlDefineErrorCode	DWORD	DW#16#0
+12.0	OdbcUserDefineErrorCode	DWORD	DW#16#0
+16.0	ODBCSqlDefineErrorSt1	STRING[6]	''
+24.0	ODBCSqlDefineErrorSt2	STRING[6]	''
+32.0	ODBCSqlDefineErrorSt3	STRING[6]	''
+40.0	ODBCSqlDefineErrorSt4	STRING[6]	''
+48.0	ODBCSqlDefineErrorSt5	STRING[6]	''
+56.0	ODBCSqlDefineErrorSt6	STRING[6]	''
+64.0	ODBCSqlDefineErrorSt7	STRING[6]	''
+72.0	ODBCSqlDefineErrorSt8	STRING[6]	''
+80.0	ODBCSqlDefineErrorSt9	STRING[6]	''
+88.0	ODBCSqlDefineErrorSt10	STRING[6]	''
=96.0		END_STRUCT	

Now if this UDT was created in a DB, SQL_DISCON is parameterised with the appropriate UTD. This is followed by the transmission of the error codes to the DB. For the individual error numbers and error statements please refer to tables 7-3 to 7-6 in the appendix where you will find the necessary explanations.

The following block diagram illustrates the syntactical sequence:

Figure 6-16 Flow Chart of SQL_DISCON



7 Application Examples

7.1 The Use of the STEP 7 Example Project

The supplied STEP 7 Example Project has been laid out for the following configuration:

- SQL database installed
- Recovery of Scales_Machine Schematic in SQL database
- Parameterisation of ODBC driver
- Datenbank GUI to be installed

Attention	**It may possibly be required to insert the used communications interface WinAC RTX into the HW configuration and to then parameterise it.**

7.1.1 Structure of an Application Programme

OB100 Complete Restart

In OB100 there is only the initialisation of a flag for the “First Call“.

OB1 CYCL_EXEC

In OB1 you skip into the appropriate function which has been selected in the “control“ variable table under “calling individual function blocks “.

FC1 CALL SQL_INIT

Calling SQL_INIT.

FC2 CALL SQL_CON

Calling SQL_CON.

FC3 CALL SQL_EXEC

Calling SQL_EXEC.

FC4 CALL SQL_DISCON

Calling SQL_DISCON.

DB10 ODBC LOGIN DATA

Parameterisation of ODBC Data Source Name (DSN), User Name, Password

Note	Input of the LOGIN DATA is case sensitive!

Control

This variable table is used to start the individual calls. For a regular sequence the blocks are processed in the following order:

- CALL FLAG SQL_INIT
- CALL FLAG SQL_CON
- CALL FLAG SQL_EXEC
- Selection of SQL statement to be transferred
  - CALL SELECT
  - CALL UPDATE
  - CALL INSERT
  - CALL DELETE
- CALL FLAG SQL_DISCON

After SQL_CON, SQL_EXEC may be called as often as required in order to send off various SQL statements.

To process an SQL_EXEC call you must first select an SQL statement.

7.1.2 Sending of a SELECT Statement using the variable table “Control “

- Open variable table “Control “ in SIMATIC MANAGER
- When "SQL_INIT First Call" is TRUE, DLL and DBIs are initialised by SQL-FBs, when it is FALSE, only DBIs are initialised.
- Set CALL FLAG SQL_INIT
- Set CALL FLAG SQL_CON
- Select CALL SELECT
- Set CALL FLAG SQL_EXEC
  - SELECT statement is sent off; data is received
- Set CALL FLAG SQL_DISCON

7.2 Adaptation of STEP 7 Example to User’s Own Requirements

7.2.1 Other SQL statements than in example project

If other SQL statements are required which are not in the example project, the user needs to make the following changes:

- The SQL strings must be adapted to the respective usages.
- The UDT’s in Argument-DB must match the placeholders in the SQL string.

The pointer transfers (lengths of data to be transferred) must match the data lengths in the DB

8 Error Messages

The WinAC SQL-DB Driver can supply different classes of error messages:

- Code in FB output STATUS_ODK_CREA_CON / STATUS_ODK_CON in accordance with WinAC-ODK
- Special error messages of SQL-DB driver

Error Type	Error Table
Odk Function Return	Odk Error Code
Odbc Function Number	Function Code
OdbcSqlDefineErrorCode	ODBC Error Code
ODBCSqlDefineErrorStxx	SQL Statements

8.1 Error Messages of WinAC ODK 4.1

The WinAC SQL-DB driver was developed with WinAC ODK (Open Development Kit). ODK can also generate error codes which are returned in **OdkSqlErrorMsg** of the FBs.

8.1.1 Error Messages for SFB65001 → STATUS_ODK_CREA_CON

These error messages can only be returned by FB **SQL_INIT**.

Table 8-1 WinAC ODK Error messages fof STATUS_ODK_CREA_CON

Code	Symbol	Description
0	NO_ERRORS	Success
0x807F	ERROR_INTERNAL	An internal error occurred.
0x8001	E_EXCEPTION	An exception occurred.
0x8102	E_CLSID_FAILED	The call to CLSIDFromProgID failed.
0x8103	E_COINITIALIZE_FAILED	The call to CoInitializeEx failed.
0x8104	E_CREATE_INSTANCE_FAILED	The call to CoCreateInstance failed.
0x8105	E_LOAD_LIBRARY_FAILED	The library failed to load.
0x8106	E_NT_RESPONSE_TIMEOUT	A Windows response timeout occurred.
0x8107	E_INVALID_OB_STATE	Controller is in an invalid state for scheduling an OB.
0x8108	E_INVALID_OB_SCHEDULE	Schedule information for OB is invalid.
0x8109	E_INVALID_INSTANCEID	Instance ID for SFB65001 call is invalid.
0x810A	E_START_ODKPROXY_FAILED	Controller could not load proxy DLL.
0x810B	E_CREATE_SHAREMEM_FAILED	The WinAC controller could not create or initialize shared memory area.
0x810C	E_OPTION_NOT_AVAILABLE	Attempt to access unavailable option occurred.

8.1.2 Error Messages for SFB65002 STATUS_ODK_CON/EXEC

These error messages are returned by all FBs, except SQL_INIT.

Table 8-2 WinAC ODK Error Messages for STATUS_ODK_CON/EXEC

Error Code	Symbol	Description
0	NO_ERRORS	Success
0x807F	ERROR_INTERNAL	An internal error occurred.
0x8001	E_EXCEPTION	An exception occurred.
0x8002	E_NO_VALID_INPUT	Input: the ANY pointer is invalid.
0x8003	E_INPUT_RANGE_INVALID	Input: the ANY pointer range is invalid.
0x8004	E_NO_VALID_OUTPUT	Output: the ANY pointer is invalid.
0x8005	E_OUTPUT_RANGE_INVALID	Output: the ANY pointer range is invalid.
0x8006	E_OUTPUT_OVERFLOW	More bytes were written into the output buffer by the extension object than were allocated.
0x8007	E_NOT_INITIALIZED	ODK system has not been initialized: no previous call to SFB65001 (CREA_COM).
0x8008	E_HANDLE_OUT_OF_RANGE	The supplied handle value does not correspond to a valid extension object.
0x8009	E_INPUT_OVERFLOW	More bytes were written into the input buffer by the extension object than were allocated.

8.2 Special Error Messages of SQL-DB Driver

In addition to the general error bit of the FBs several special error codes and error statements are supplied in OdkSqlErrorMsg, which describe the cause in more detail.

8.2.1 ODK-Function Returns

Table 8-3 ODK-Function Returns

ODK-Error-Code	Function Description
9500	Wrong format specification in STRING for STATEMENT
9501	Read S7 STRING failed
9502	Buffer Overflow in STATEMENT read
9503	Read S7 ARGUMENT TYPE failed
9504	Read S7 ARGUMENT failed by INT
9505	Read S7 ARGUMENT failed by SMINT
9506	Read S7 ARGUMENT failed by DOUBLE
9507	Read S7 ARGUMENT failed by REAL
9508	Read S7 ARGUMENT failed by STRING
9509	Wrong S7 ARGUMENT TYPE not defined
9510	No PRINTF ARGUMENT in STRING for reserved ARGUMENT

9511	Read ODBC LOGIN PARA failed
9512	ODBC LOGIN PARA copy failed
9513	Write ODK SQL ERRORBIT failed
9514	Write ODK SQL BUSYBIT failed
9515	Set DATA EVENT failed
9516	Cannot create DATA EVENTHANDLE
9517	Data event not yet ready
9518	WAIT FOR SINGLE OBJ failed data event
9519	ODK DATA EVENT eradication failed
9520	BUFFEROVERFLOW in value write
9521	READ S7 VALUETYPE failed
9522	Write S7 VALUE failed by INT
9523	Write S7 VALUE failed by SMINT
9524	Write S7 VALUE failed by DOUBLE
9525	Write S7 VALUE failed by REAL
9526	Write S7 VALUE failed by STRING
9527	Wrong S7 VALUE TYPE not defined
9528	No S7 VALUETYPE for SQLVALUETYPE available
9529	Non conforming VALUETYPE from SQLSERVER to S7 DATABLOCK, expected INT
9530	Non conforming VALUETYPE from SQLSERVER to S7 DATABLOCK, expected SMINT
9531	Non conforming VALUETYPE from SQLSERVER to S7 DATABLOCK, expected DOUBLE
9532	Non conforming VALUETYPE from SQLSERVER to S7 DATABLOCK, expected REAL
9533	Non conforming VALUETYPE from SQLSERVER to S7 DATABLOCK, expected STRING
9534	Event STANDBY TIME too long, no REQUEST from ASYNC THREAD
9535	Set INIT EVENT failed
9536	Cannot create ODBC INIT EVENTHANDLE
9537	CLOSE DATA EVENT failed
9538	Not defined WAITFORSINGLEOBJ RETURN in DATA EVENT
9539	ODK ODBC INIT EVENT eradication failed
9540	CLOSE ODBC INIT EVENT failed
9541	ODBC INIT EVENT not yet ready
9542	WAIT FOR SINGLE OBJ failed for ODBC INIT EVENT
9543	Not defined WAITFORSINGLEOBJ RETURN for ODBC INIT EVENT
9544	ODBC INIT EVENT timeout
9545	EXEC EVENT timeout
9546	Read timeout TIMEODBCEVENT from S7 failed
9547	Read TIMEOUT TIMEEXEC from S7 failed
9548	Cant create ODBC CLOSEDB EVENTHANDLE
9549	ODBC CLOSEDB EVENT timeout
9550	ODK ODBC CLOSEDB EVENT eradication failed
9551	Close ODBC CLOSEDB EVENT failed

9552	ODBC INIT CLOSEDB not yet ready
9553	WAIT FOR SINGLE OBJ FAILED for ODBC CLOSEDB EVENT
9554	Not defined WAITFORSINGLEOBJ RETURN for ODBC CLOSEDB EVENT
9555	Not defined PRINTF ARGUMENT in STRING
9556	Cannot find DLL PATH and FILENAME
9557	Cannot read VERSION INFO SIZE
9558	Cannot read VERSION INFO
9559	Cannot verify query the value
9560	Version Nr. does not match, different versions STEP 7 to DLL

8.2.2 Function-Code Numbers

Table 8-4 Function-Code Numbers

Function Code	Function Description
9000	SQL ALLOC ENVIRONMENT HANDLE
9001	ODBC VERSION ENVIRONMENT ATTRIBUTE
9002	SQL ALLOC CONNECTION HANDLE
9003	CONNECT DATA SOURCE
9004	SQL ALLOC ARGUMENT HANDLE
9005	BUILT SQL STRING
9006	COUNT OF COL
9007	DESCRIBE COL TYPE
9008	BIND COLUMN WHITH VALUETYPE
9009	FETCH ARGUMENT
9010	FREEING ARGUMENT HANDLE
9011	CLOSE CONECTION
9012	FREEING CONNECTION HANDLE
9013	FREEING ENVIRONMENT HANDLE

8.2.3 ODBC-Function Errors

Table 8-5 ODBC-Function Errors

ODBC-Error-Code	ODBC-Error Description
0000	ODBC FUNCTIONRET ok
8501	ODBC SQL OBJECT CALL initialization failed
8502	ODBC SQL OBJECT call execution failed
8503	ODBC SQL OBJECT call FETCH failed
8504	ODBC SQL OBJECT call CLOSE DATABASE failed
8505	ODBC FETCH ODK buffer Overflow
8506	undefined sql value type during create column type
8507	Sql unknown database column type
8508	Something wrong with sql database in character set

8509	No data existing
8510	Error in SQLGETDIAGREC
8511	Too long STRING CHAR in SQL DB
8512	Database already connected
8550	Success with info
8551	No data
8552	Still executing
8553	Need data

8.2.4 SQL-Statements

Table 8-6 SQL-Statements

SQLSTATE	Error
01000	General warning
01001	Cursor operation conflict
01002	Disconnect error
01003	NULL value eliminated in set function
01004	String data, right truncated
01006	Privilege not revoked
01007	Privilege not granted
01S00	Invalid connection string attribute
01S01	Error in row
01S02	Option value changed
01S06	Attempt to fetch before the result set returned the first row set
01S07	Fractional truncation
01S08	Error saving File DSN
01S09	Invalid keyword
07001	Wrong number of parameters
07002	COUNT field incorrect
07005	Prepared statement not a *cursor-specification*
07006	Restricted data type attribute violation
07009	Invalid descriptor index
07S01	Invalid use of default parameter
08001	Client unable to establish connection
08002	Connection name in use
08003	Connection does not exist
08004	Server rejected the connection
08007	Connection failure during transaction
08S01	Communication link failure
21S01	Insert value list does not match column list
21S02	Degree of derived table does not match column list
22001	String data, right truncated

22002	Indicator variable required but not supplied
22003	Numeric value out of range
22007	Invalid datetime format
22008	Datetime field overflow
22012	Division by zero
22015	Interval field overflow
22018	Invalid character value for cast specification
22019	Invalid escape character
22025	Invalid escape sequence
22026	String data, length mismatch
23000	Integrity constraint violation
24000	Invalid cursor state
25000	Invalid transaction state
25S01	Transaction state
25S02	Transaction is still active
25S03	Transaction is rolled back
28000	Invalid authorization specification
34000	Invalid cursor name
3C000	Duplicate cursor name
3D000	Invalid catalog name
3F000	Invalid schema name
40001	Serialization failure
40002	Integrity constraint violation
40003	Statement completion unknown
42000	Syntax error or access violation
42S01	Base table or view already exists
42S02	Base table or view not found
42S11	Index already exists
42S12	Index not found
42S21	Column already exists
42S22	Column not found
44000	WITH CHECK OPTION violation
HY000	General error
HY001	Memory allocation error
HY003	Invalid application buffer type
HY004	Invalid SQL data type
HY007	Associated statement is not prepared
HY008	Operation canceled
HY009	Invalid use of null pointer
HY010	Function sequence error
HY011	Attribute cannot be set now

HY012	Invalid transaction operation code
HY013	Memory management error
HY014	Limit on the number of handles exceeded
HY015	No cursor name available
HY016	Cannot modify an implementation row descriptor
HY017	Invalid use of an automatically allocated descriptor handle
HY018	Server declined cancel request
HY019	Non-character and non-binary data sent in pieces
HY020	Attempt to concatenate a null value
HY021	Inconsistent descriptor information
HY024	Invalid attribute value
HY090	Invalid string or buffer length
HY091	Invalid descriptor field identifier
HY092	Invalid attribute/option identifier
HY095	Function type out of range
HY096	Invalid information type
HY097	Column type out of range
HY098	Scope type out of range
HY099	Nullable type out of range
HY100	Uniqueness option type out of range
HY101	Accuracy option type out of range
HY103	Invalid retrieval code
HY104	Invalid precision or scale value
HY105	Invalid parameter type
HY106	Fetch type out of range
HY107	Row value out of range
HY109	Invalid cursor position
HY110	Invalid driver completion
HY111	Invalid bookmark value
HYC00	Optional feature not implemented
HYT00	Timeout expired
HYT01	Connection timeout expired
IM001	Driver does not support this function
IM002	Data source name not found and no default driver specified
IM003	Specified driver could not be loaded
IM004	Driver's **SQLAllocHandle** on SQL_HANDLE_ENV failed
IM005	Driver's **SQLAllocHandle** on SQL_HANDLE_DBC failed
IM006	Driver's **SQLSetConnectAttr** failed
IM007	No data source or driver specified; dialog prohibited
IM008	Dialog failed
IM009	Unable to load translation DLL

IM010	Data source name too long
IM011	Driver name too long
IM012	DRIVER keyword syntax error
IM013	Trace file error
IM014	Invalid name of File DSN
IM015	Corrupt file data source

9 List of Abbreviations

DB	Data block
FB	Function block
OB	Organisation block
RTX	Real Time eXtension for Windows
UDT	User defined type (data type definition in STEP 7)

10 History

Table 10-1

Version	Date	Remark
V1.20	11-02-09	Tested with WinAC RTX 2009