DbDrv driver of the REX control system
User guide

REX Controls s.r.o.

Version 2.50.5
Plzeň (Pilsen), Czech Republic
2017-09-06

Contents

Chapter 1

The DbDrv driver and the REX Control System

1.1 Introduction

This manual describes the DbDrv driver for connecting to arbitrary SQL database, for which an ODBC interface exist on the target platform. The driver allows both reading and writing data. Further it is possible to export the archives of the REX Control System to the database, i.e. alarms, events and trends.

1.2 System requirements

The DbDrv driver can be used on Windows and Linux target devices. A TCP/IP stack is required for communication, Ethernet card or USB WiFi dongle must be available in the system. An ODBC driver must be installed on the target device.

In order to use the driver, the host computer (development) and the target computer (runtime) must have the following software installed:

Development computer	
Operating system	one of the following: Windows 7/8/10
Development tools	version of the REX Control System development tools for Windows operating system

Target device	
REX runtime core	version for the corresponding operating system
IO driver	version for the corresponding operating system
ODBC interface	version for the corresponding database (MySQL, PostgreSQL, MS-SQL, ...)

1.3 Installation of the driver on the host computer

The DbDrv driver is included in the installation package of the Development tools of the REX Control System. It is necessary to select the corresponding package in the installer. The REX Control System typically installs to the
C:\Program Files\REX Controls\REX_<version> folder.

The following files are copied to the installation folder:

bin\DbDrv_H.dll – Configuration part of the DbDrv driver.

bin\DbDrv_T.dll – Target part of the DbDrv driver which is called by the RexCore runtime module.

DOC\ENGLISH\DbDrv_ENG.pdf – This user manual.

1.4 Installation of the driver on the target device

1.4.1 Windows machines

The target part of the driver, which is used for connecting to the database is included in the Development tools of the REX Control System as mentioned above.

1.4.2 Linux machines

If there is no RexCore runtime module installed on your target device, install it first using the Getting started guide of the REX Control System for the given platform, e.g. [1].

In order to enable connection to SQL databases from the REX Control System the driver must be installed. This is done from command line using the command
Debian Linux:
`sudo apt-get install rex-dbdrvt`

1.5 Installation of ODBC interface on the target device

It is necessary to install the ODBC interface for the corresponding database system on target device on any operating system.

On Debian Linux, the `rex-dbdrvt` package automatically install and configure ODBC interface to be used with connection string mode for:

- MySQL (package `libmyodbc`), ODBC driver name `MySQL`,
- Microsoft SQL (MSSQL) (package `tdsodbc`), ODBC driver name `MSSQL`,
- PostgreSQL (package `odbc-postgresql`), ODBC driver name `PostgreSQL`.

No more manual configuration is required. Following information about particular database system connection details is usually not needed.

1.5.1 Debian Linux – MySQL

The necessary packages are `unixodbc` and `libmyodbc`. Install them using:
`sudo apt-get install unixodbc libmyodbc`

Further it is necessary to append the following section to the `/etc/odbcinst.ini` file:

```
[MySQL]
Description     = MySQL driver
Driver          = libmyodbc.so
Setup           = libodbcmyS.so
```

1.5.2 Debian Linux – Microsoft SQL (MSSQL)

The necessary packages are `unixodbc` and `tdsodbc`. Install them using:
`sudo apt-get install unixodbc tdsodbc`

Further it is necessary to append the following section to the `/etc/odbcinst.ini` file:

```
[MSSQL]
Description     = Microsoft SQL driver
Driver          = libtdsodbc.so
Setup           = libtdsS.so
```

It is recommended to allways define PORT value in connection string with Microsoft SQL driver, because the default value can vary with ODBC adapter build configuration.

1.5.3 Debian Linux – Common ODBC DSN Configuration

Optionally, connection parameters can be stored under a specified name – a *DSN* – in the `/etc/odbc.ini` file:

```
[MyDSN]
Driver       = MSSQL
Description  = Microsoft SQL server - My great application
SERVER       = sqlsrv.example.com
PORT         = 1433
Database     = MyDatabase
```

Then the *connection-string* have to be specified in form:
`DSN=MyDSN;UID=username;PWD=password;`.

Putting username and password into DSN configuration in `/etc/odbc.ini` is usually not supported (depends on database system driver).

1.5.4 Other platforms and database systems

Instructions on installing the ODBC driver on your platform should be included in the documentation of your database system (MySQL, PostgreSQL, Microsoft SQL etc.).

Chapter 2

Including the driver in the project

The driver is included in the project as soon as the driver is added to the project main file and the inputs and outputs are connected in the control algorithms.

2.1 Adding the `DbDrv` driver

The project main file with the `DbDrv` driver included is shown in Figure 2.1.

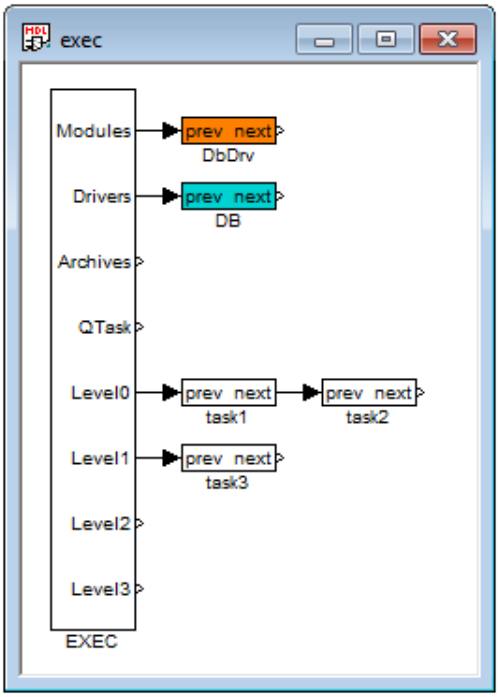

Figure 2.1: An example of project main file with the `DbDrv` driver included

There are 2 blocks which must be added to the project to include the driver. First the `MODULE` block is attached the the `Modules` output of the `EXEC` function block. It must be renamed to `DbDrv`.

The other block named `DB` is of type `IODRV` and it is connected to the `Drivers` output of the main `EXEC` block. The three most important parameters are:

`module` – name of the module linked to the driver, in this case `DbDrv` – the name is CASE SENSITIVE!

`classname` – class of the driver, `DbDrv` in this particular case
The name is CASE SENSITIVE!

`cfgname` – name of the driver configuration file (`*.rio`, REX Input/Output), which is discussed in chapter 3

The name of this block (`DB`, see Fig. 2.1), is the prefix of all input and output signals provided by this driver.

The above mentioned parameters of the `IODRV` function block are configured in the RexDraw program as shown in Figure 2.2.

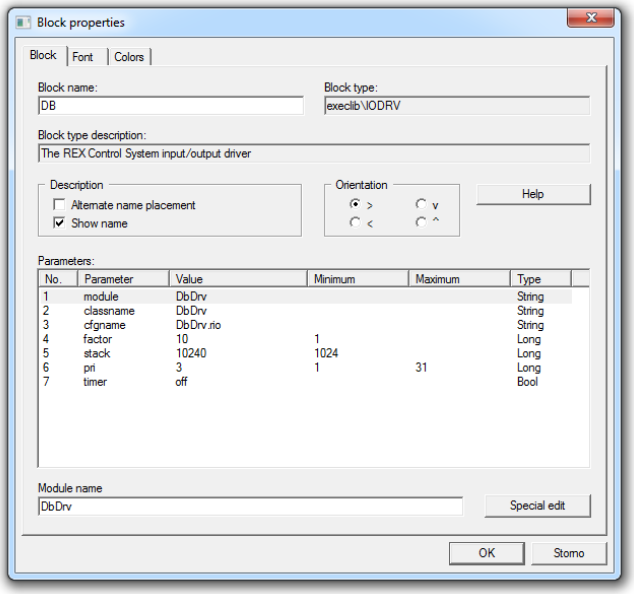

Figure 2.2: Settings of the `DbDrv` I/O driver

Chapter 3

Driver configuration

For the driver configuration, there is a comprehensive configuration dialog available. It can be opened by pressing the `Special edit` button in driver instance `IODRV` block parameters dialog. The resulting configuration is stored in `*.rio` file as standard for other REXdrivers.

Documentation for the configuration dialog is not yet completed. But its structure is well organized and easily understoodable. Here we provide some screenshots of the dialog – figures 3.1, 3.2, 3.3. For details about particular function modes, plese refer to configuration file format description in chapter 3.1.

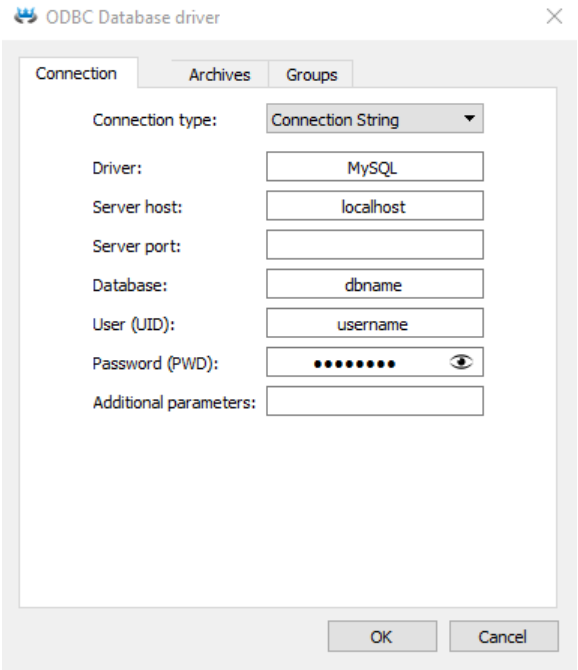

Figure 3.1: Configuration dialog of `DbDrv` driver – Database connection details

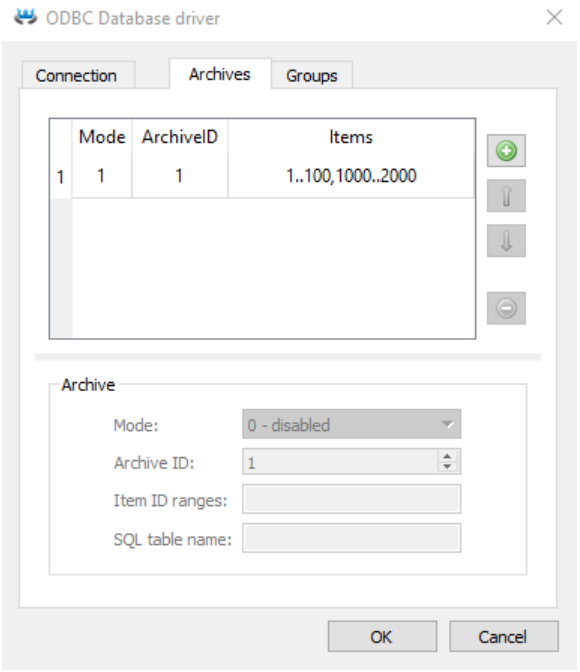

Figure 3.2: Configuration dialog of `DbDrv` driver – Archives section configuration

3.1 Format of the configuration file

The `*.rio` file is a text file, therefore it can be edited in any plain-text editor (e.g. Notepad). The structure is illustrated below:

```
ODBC {
  Connection "DRIVER=MySQL;SERVER=192.168.1.200;PORT=3306;"
"DATABASE=dbname;UID=dbuser;PWD=dbpassword;"
  #Connection "DSN=mydatabase;"
  Archive {
   Mode      1
   SQL       "alarm1"
   ArchiveID 0
   Items     0,104,114,1000
  }
  #Archive {
   Mode      2
   SQL       "trend1"
   ArchiveID 0
   Items     100,100
  }
```

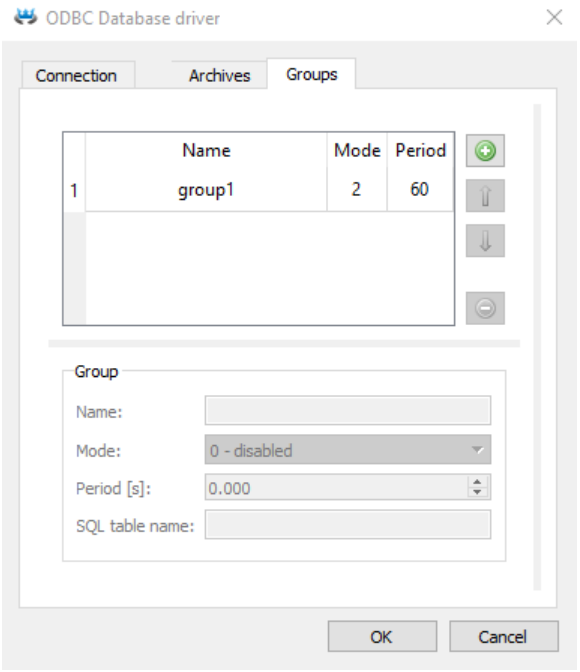

Figure 3.3: Configuration dialog of DbDrv driver – Groups section configuration

```
  Archive {
   Mode      3
   SQL       "insert into trend1 (Time, GroupID, Value1, Value2, Value4)"
"values(?T,?I,?1,?,?4)"
   ArchiveID 0
   Items     100,111
  }
  Group {
   Mode      2
   SQL       "test1"
   Period    5
   Items     "r1,i2,i3,r4"
  }
  Group {
   Mode      130
   SQL       "test2"
   Period    60
   Items     "d1,d2,i10,i11"
  }
  Group {
   Mode      3
```

```
   SQL       "select d1,d2,i10,i11, Time from test2 where Time < ? and "
"StationID=000 order by Time desc,ID desc limit 2"
   Period    5
   Items     "gr1,gr2,gr3,gr4"
  }
  Group {
   Mode      131
   SQL       "insert into test2 (Time,StationID,d1,d2,i10,i11) values "
"(UTC_TIMESTAMP(),2,?1,?,?,?)"
   Period    10
   Items     "gw1,gw2,gw3,gw4"
  }
}
```

The parameters whose name start with `#` are ignored and can be used for comments. Long lines (typically SQL queries) can be split among several lines as shown above. There can be no white-space characters at the beginning of consecutive lines. The `Alarm` section is repeated as necessary to define all the rules (filters) for exporting archives of the REX Control System. Similarly for reading and writing data directly from the algorithm the `Group` sections are used. All the parameter and section names are case sensitive.

The meaning of individual parameters follows:

`Connection` – The so-called *connection-string* defining the database to connect to. A full *connection-string* with all the parameters can be used. Alternatively it is possible to define the connection by DSN (DataSourceName), which is defined within the ODBC interface and contains all the necessary information for the connection.

`Mode` – Defines the structure of reading/writing from/to the database. The options are:

For the `Archive` section:

- 0 Nothing gets exported (used for disabling the item temporarily).
- 1 Only the alarms and events are exported (filtered by additional parameters). The table in the database must contain the following columns: `Time`, `AlarmID`, `Code`, `Level`, `Value`.
- 2 Only the trends are exported, i.e. the data stored by the `TRND` block. The data is filtered by additional parameters. The table in the database must contain the following columns: `Time`, `GroupID`, `Value1`, `Value2`, ...
- 3 Only the trends are exported, but on the contrary to the above the `SQL` parameter has the meaning of a full SQL query, to which the values are injected. The following placeholders can be used: ?T = time; ?I =itemID; ?1 =1st item; ?2 =2nd item; ... A plain question mark has the meaning of *next item* in the following order: 1st value, 2nd value, ...;)

For the `Group` section:

0 Nothing gets read (used for disabling the item temporarily).

1 It is assumed that the table is ordered by the `ID` column. The row with the highest ID is supplied to the corresponding input flags in the task. The columns and the items/flags must have the same name.

2 It is assumed that the table is ordered by the `Time` column (`ID` is the secondary key). The row with the highest time not placed in the future is selected and the resulting data is supplied to the corresponding input flags in the task. The columns and the items/flags must have the same name. This mode allows applying a pre-generated sequence of data.

3 The SQL query from the `SQL` parameter is executed, the inputs are updated by the first row of the response (1st column to the 1st item, ...). It is possible to use ?T in the SQL query, which gets replaced by the current time. It is also possible to use ?1 in the SQL query, which gets replaced by the value of 1st item, ?2 , which gets replaced by the value of 2st item,...

128 Nothing gets written (used for disabling the item temporarily).

129 The values from the corresponding flags in the tasks are written to the database. The columns and the items/flags must have the same name.

130 Similar to the above, only there is one more column named `Time` which contains the current time of the REX runtime core in UTC.

131 The SQL query from the `SQL` parameter is executed. The following placeholders can be used: ?T = time; ?I =itemID; ?1 =1st item; ?2 =2nd item; ... A plain question mark has the meaning of *next item* in the following order: 1st value, 2nd value, ...;)

`SQL` – Table name or full SQL command as defined by the `Mode` parameter. In some cases the notation `<database_name>.<table_name>` must be used for referencing database tables.

`ArchiveID` – Number of archive to read the data from. The archives are numbered from 1 in the order of appearance in the configuration of REX executive (`ARCHIVE` blocks connected to the `EXEC` block).

`Period` – Period in seconds to generate the SQL query.

`Items` – A list of items to read/write (in the `Group` section) or a range of IDs (the `id` parameter of the originating block) to export from archive to the database (in the `Archive` section). In the case of archives there must be even number of entries, where the odd entries define the start of an interval and the even entries define its end. Therefore e.g. "100,100,104,109" means IDs 100, 104, 105, ..., 109. The entries must be sorted in ascending order.

3.2 Connecting the inputs and outputs in the control algorithm

The inputs and outputs of the driver must be interconnected with the individual tasks (`.mdl` files). The individual tasks (`QTASK` or `TASK` blocks) are connected to the `QTask`, `Level0`,..., `Level3` outputs of the main `EXEC` block. Use the blocks depicted in Fig. 3.4 to interchange data between the control algorithm and the `DbDrv` driver.

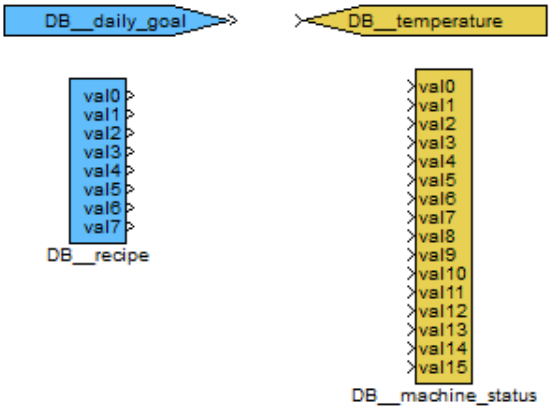

Figure 3.4: Example of input and output flags of the `DbDrv` driver

The `From` block allowing the user to read one input signal has the `Goto tag` set to `DB__<IN>`. The `Goto` block allowing the user to set one output signal has the `Goto tag` set to `DB__<OUT>`, where `<IN>` and `<OUT>` are strings referring to the items defined in the `*.rio` configuration file. All the strings used for accessing data provided or accepted by the driver always have the `DB` prefix right at the beginning of the tag followed by two mandatory `_` characters (underscore).

The rest of the input or output string reference is interpreted by the driver as defined in the `*.rio` configuration file.

There are additional auxiliary signals for each I/O signal. These can be assessed by appending the following strings to the signal reference:

`_Disable` – If `True`, the read/write operations for the whole group are disabled.

`_Trigger` – A rising edge triggers execution of the read/write operation.

`_Age` – Number of seconds since the last read/write database access.

`_Fresh` – Same as `_Age`

`_AgeDb` – Number of seconds since the last read/write database access, on the contrary to the above the age is defined by the database item.

`_FreshDb` – Same as `_AgeDb`

`_Period` – Contains or sets the `Period` parameter, i.e. the period of SQL query execution as defined for each group.

There are additional auxiliary global signals:

`Connect` – connection to database server.

`Connected` – status of the connection to database server.

`Reset` – reseting archive reading (all archiving groups).

`Resetting` – pushed off when reseting is finished.

The driver supports multi-flags, therefore it is possible to read/write several signals at once as displayed in Figure 3.4. See the `INQUAD`, `OUTQUAD`, `INOCT`, `OUTOCT` and `INHEXD`, `OUTHEXD` function blocks [2]. In this case the block name references the first object and the signals are mapped to this object and the consecutive ones (groups of 4, 8 or 16). This preserves communication bandwidth and also clarity of the algorithm.

Chapter 4

Implementation details

Additional information about the use and implementation of the `DbDrv` driver in the REX Control System is gathered in this chapter.

- The `Items` parameter in the `Archive` section is a list of numbers, where the odd entries mean *from* and the even ones *to*. E.g. `Items "2, 5, 10, 15"` exports items with IDs 2 to 5 and 10 to 15. There must be even number of entries even if exporting items with only one ID. The entries in the `Items` parameter must be sorted in ascending order.
- Hovever the majority of database systems is case-insensitive, the control system REX is case-sensitive. Therefore the `DbDrv` driver is also case-nsensitive in the I/O flags (the flags correspond with column names in the database).
- All values written or read to/from the database are decimal numbers (type double). The database columns can be of other type because SQL queries are textual. The optional parametr `Type` can be used, where `i` means the value is processed as `long` type, `b` denotes a `bool` type, `s` denotes a `string` type and `r` denotes a `real`. For example `Type "rrisb"` means 1st and 2nd values are real number, 3th value is integer, 4th value is string and 5th value is boolean.
- The flags must be unique in the whole project because they contain no `Group` identifier. Only the first occurrence is processed in the case of duplicities.
- It is possible to define (optional) parameter `Name` in the `Group` section. The flags must be in form `<group name>_<item name>"` in this case.
- The driver need username and pasword for login into database. Both is stored in *.rio file in plain text as all other parameters. Therefore it is strongly recommended using dedicated login name with very restricted permisions.
- All timestamps (e.g. substitute for `?T` in SQL queries) are expanded into string in form `<year>-<month>-<day> <hour>:<minute>:<second>.<mikrosecond>`. The UTC timezone is used.

- Using SQL data type with at least microsecond resolution is recommended for timestamps.
  - MySQL: `DATETIME(6)` [1]
  - Microsoft SQL (2008+): `datetime2` [2]
  - PostgreSQL: `timestamp` [3]
- Current implementation limits SQL string to 1023 characters (after expansion question marks). The parameter `Items` (in both `Archive` section and `Group` section) is limited to 64 values.
- The column `Code` in alarm export is integer number where lowest 5bit is alarm class and higer 3 bits is alarm subtype. The classes are:

0	System alarm
1	Bool alarm
2	Byte value alarm
3	Short value alarm (signed 16 bits integer number)
4	Long value alarm (signed 32 bits integer number)
5	Word value alarm (unsigned 16 bits integer number)
6	DWord value alarm (unsigned 32 bits integer number)
7	Float value alarm
8	Double value alarm
10	Large value alarm (signed 64 bits integer number)
12	String value alarm
13 .. 16	Not used
17	Bool value group event
18	Byte value group event
19	Short value group event
20	Long value group event
21	Word value group event
22	Dword value group event
23	Float value group event
24	Double value group event
26	Large value group event
31	Alarm acknowledge

  The subtypes for system alarms are:

0	Date mark (is not exported)
1	Executive event
2	Archive event

[1] http://dev.mysql.com/doc/refman/5.7/en/fractional-seconds.html
[2] https://msdn.microsoft.com/en-us/library/bb677335.aspx
[3] https://www.postgresql.org/docs/current/static/datatype-datetime.html

The level indicate event in this case. The executive events are:

0 System reset
1 Download begin
2 Download end
3 Download failed
4 Executive stop
5 Executive start
6 Executive swap
7 Time set

and the archive event:

0 Archive cleared (not used now)
1 Archive rekonstruction saved (not used now)
2 Archive rekonstruction normal (not used now)
3 Checksum error (not used now)
4 Integrity error (not used now)
5 Sizes changed (not used now)
6 Limit exceed (disk archives only)
7 Buffer overflow

The subtypes for boolean alarms are:

0 High to Low (e.g. the attached boolean variable have been changed from the high/true/1 value to the low/false/0 value)
1 Low to High (e.g. the attached boolean variable have been changed from the low/false/0 value to the high/true/1 value)

The subtypes for numeric alarms are:

0 low alarm
1 high alarm
2 2nd low alarm
3 2nd high alarm

The level 0 indicate end of alarm conditions. The alarms with level from 128 to 255 not indicate end of alarm conditions by this special alarm event. The alarm acknowledge should has same subtype and level as acknowledged alarm. However not active alarms are regarded as acknowledged by the acknowledge of any subtype. The 1st level alarm is also regarded as acknowledge by the acknowlwdge of the 2nd level.

Chapter 5

Troubleshooting

Just like in the case of any other problem it is recommended to view the error and debug information (the System Log tab in RexView). Unsuccessful database connections and/or misconfigured SQL queries are listed in the log. The most frequent problems include:

- The ODBC interface for the corresponding database is not installed or correctly configured on the device (the odbcinst.ini and odbc.ini files in Linux).
- Invalid database *connection-string*.
- The requested tables are not available in the database. The database might be case sensitive.
- Inconsistency in naming of the columns. The database might be case sensitive.
- Although the `DbDrv` driver uses very simple SQL syntax, there are some differences among individual database systems.
- Especially when defining the SQL queries by hand, it is necessary to double-check the syntax.
- Duplicated item name in the `Items` parameter, which results in the item being not available.

In the case that the given input or output works with other software tools and does not work in the REX Control System, report the problem to us, please. E-mail is preferred, reach us at `support@rexcontrols.com`. Please include the following information in your description to help us process your request as soon as possible:

- Identification of the REX Control System you are using. Simply export it to a file using the RexView program (Target → Licence → Export).
- Short and accurate description of your problem.
- The configuration files of the REX Control System (`.mdl` files) reduced to the simplest case which still demonstrates the problematic behavior.

Bibliography

[1] REX Controls s.r.o.. *Getting started with REX on Raspberry Pi*, 2017.

[2] REX Controls s.r.o.. *Function blocks of the REX Control System – reference manual*, 2017.

Documentation reference number: 8614