Function block library

DBFL_SQL_5

for PLCnext Engineer

Documentation for
PHOENIX CONTACT function blocks
PHOENIX CONTACT GmbH Co. KG
Flachsmarktstrasse 8
D-32825 Blomberg, Germany

This documentation is available in English only.

Table of Contents

1 Installation hint

If you did not specify a different directory during **library** installation all data in the MSI file will be unpacked to

c:\Users\Public\Documents\Phoenix Contact Libraries\PLCnext Engineer (former: PC Worx Engineer)

Please copy the library data to your PLCnext Engineer (former: PC Worx Engineer) working library directory.

If you did not specify a different directory during **PLCnext Engineer** installation the default PLCnext Engineer working library directory is

c:\Users\Public\Documents\PLCnext Engineer\Libraries (former: PC Worx Engineer\Libraries)

2 General information

This library offers function blocks as database drivers for MS SQL, MySQL and MariaDB applications.

2.1 Introduction

This user manual describes the DBFL function block group for the MSSQL and MySQL / MariaDB libaries. Both libraries extend the scope of functions of the PC Worx automation software from Phoenix Contact. Depending on which SQL dialect is used, the corresponding library should be used.

The AXC F 2152 (2404267) is connected to the SQL database via Ethernet. Input and output commands of the function block are written to the SQL database as data records, where they are stored and can be called later. Using the DBFL function blocks, the project is programmed in PC Worx.

2.2 Structure of this manual

Section 1 contains basic information about using the DBFL function blocks.

Section 2 provides information about the individual function blocks and their input and output parameters.

Section 3 uses examples to show how data is written to and stored in an SQL database using the DBFL function blocks (for the relevant SQL dialect).

Section 4 provides an overview listing the corresponding pages for the individual function blocks in the manual.

For information about operating PC Worx, please refer to the "PC Worx Quick Start Guide".

Validity

This manual is valid for the DBFL_BootToCom_T1 to DBFL_WordToComT2 function blocks.

2.3 Documentation

Latest documentation

Make sure you always use the latest documentation. Find out from the manufacturer whether any changes or additions have been made to the documentation used.

Additional documentation Please note the information about the following in the user manuals:

- The control system used
- The development environment used
- The I/O devices (e.g., sensors/actuators, etc.) that are connected to the control systems and the function block

2.4 Abbrevations and symbols

MicrosoftSQL Server 20xx

MySQL Version 5.x

MariaDB 10.4.7-GA

2.5 System requirements

Hardware

- A PC
- An AXC F 2152 (2404267)
- An Ethernet connection between the PC and the controller used

Software

- Operating system: Windows XP Professional SP2
- DOT.NET Framework 2.0, including current service pack
- PC WorX Engineer
- DBFL_SQL_1
- SQL Server 20xx
- DBFL_SQL_1
- MySQL Community Server 5.x.x
- MySQL GUI Tools
- MariaDB (HeidiSQL)

NOTE: The rights concept for the database must be clarified with the system administrator and implemented. PHOENIX CONTACT does not accept any liability for damage caused as a result of inappropriate use of the function blocks.

Knowledge of SQL (SQL=Structured Query Language) is required in order to set up the DBFL function blocks. Knowledge of the "Data Manipulation Language" (DML) is essential:

Data Manipulation Language (DML)

The Data Manipulation Language (DML) is the core component of SQL. It enables controlled data access and modification. The Data Manipulation Language includes four SQL commands: INSERT, SELECT, UPDATE, and DELETE. These four basic commands can be used to manage data in SQL databases:

- INSERT: Inserts new data
- SELECT: Filters data from a database
- UPDATE: Modifies existing data records
- DELETE: Deletes selected data records from the database

3 Change notes

Library version	Library build	PLCnext Engineer version	Change notes	Supported PLCs
5	20200206	>= 2020.0 LTS	Released for 2020.0 LTS	AXC F 1152 (1151412) AXC F 2152 (2404267)
5	20191001	2019.0 LTS 2019.3 2019.6 2019.9	Adapted to PLCnext Engineer 2019.9	AXC F 2152 (2404267)
4	20190830	2019.0 LTS 2019.3 2019.6	Release for MariaDB 10.4.7-GA	AXC F 2152 (2404267)
4	20190716	2019.0 LTS 2019.3 2019.6	Adapted to PLCnext Engineer 2019.6	AXC F 2152 (2404267)
3	20190225	2019.0 LTS 2019.3	Supports “Allow extended identifiers” = ON	AXC F 2152 (2404267)
3	20190219	2019.0 LTS	DBFL_SQL_3: • Adapted to PLCnext Engineer 2019.0 LTS	AXC F 2152 (2404267)
2	—	7.3	DBFL_SQL_2: • Adapted tp PLCnext Engineer 7.3	AXC F 1050 (2404701) AXC F 2152 (2404267)
1	20180814	7.2.2	DBFL_SQL_1: • Converted from PC Worx 6	AXC F 1050 (2404701) AXC F 2152 (2404267)

New version number: Functional changes of at least one function block, incompatibilities (e.g. change of library format)

New build number: No functional changes, but changes in the MSI file (e.g. documentation update, additional examples)

4 Function blocks

This manual describes the DBFL function block group. It includes:

- Database-specific function blocks
- Function blocks for internal use
- Extended function blocks

Extended function blocks

Extended function blocks can be used to compile an SQL command step-by-step. T1 and T2 are series characteristics:

Function block	Description	Version	Supported articles	License
DBFL_TSQL_ACCESS	The function block enables access to the MsSQL database.	2	-	none
DBFL_MySQL_ACCESS	The function block enables access to the MySQL / MariaDB database.	2	-	none
DBFL_TSQL_DECODE	The function block is used to evaluate a received table and is used as a continuation block of DBFL_TSQL_ACCESS.	2	-	none
DBFL_MySQL_DECODE	The function block is used to evaluate a received table and can be used as a continuation block of DBFL_MySQL_ACCESS.	2	-	none
DBFL_CommandFiFo	The function block saves up to 50 SQL commands.	1	-	none
DBFL_CODE	The function block inserts SQL commands or parts of them in the "SQL_OUT" array.	1	-	none
DBFL_StartComT1	The function block creates the start of a database command.	1	-	none
DBFL_StartComT2	The function block creates the start of a database command.	1	-	none
DBFL_BoolToComT1	The function block inserts a Boolean value in the SQL command.	1	-	none
DBFL_BoolToComT2	The function block inserts a Boolean value in the SQL command.	1	-	none
DBFL_IntToComT1	The function block inserts an integer value in the SQL command.	1	-	none
DBFL_IntToComT2	The function block inserts an integer value in the SQL command.	1	-	none
DBFL_DIntToComT1	The function block inserts a DINT value in the SQL command.	1	-	none
DBFL_DIntToComT2	The function block inserts a DINT value in the SQL command.	1	-	none
DBFL_RealToComT1	The function block inserts a REAL value in the SQL command.	1	-	none
DBFL_RealToComT2	The function block inserts a REAL value in the SQL command.	1	-	none

DBFL_DateTimeStrT1	The function block inserts a date/time value in the SQL command.	2	-	none
DBFL_DateTimeStrT2	The function block inserts a date/time value in the SQL command.	2	-	none
DBFL_ByteToComT1	The function block inserts a byte value in the SQL command.	1	-	none
DBFL_ByteToComT2	The function block inserts a byte value in the SQL command.	1	-	none
DBFL_WordToComT1	The function block inserts a data word in the SQL command.	1	-	none
DBFL_WordToComT2	The function block inserts a data word in the SQL command.	1	-	none
DBFL_StrToComT1	The function block inserts a string in the SQL command.	1	-	none
DBFL_StrToComT2	The function block inserts a string in the SQL command.	1	-	none

Extended function blocks with series characteristic T1 are processed cyclically.

Extended function blocks with series characteristic T2 are controlled via IN_xActivate.

5 DBFL_TSQL_ACCESS

The function block enables access to the database. The parameters required for the connection (DB_USER, DB_PASSWORD, IP_PORT, IP_ADDRESS, DB_NAME) must be a STRING.

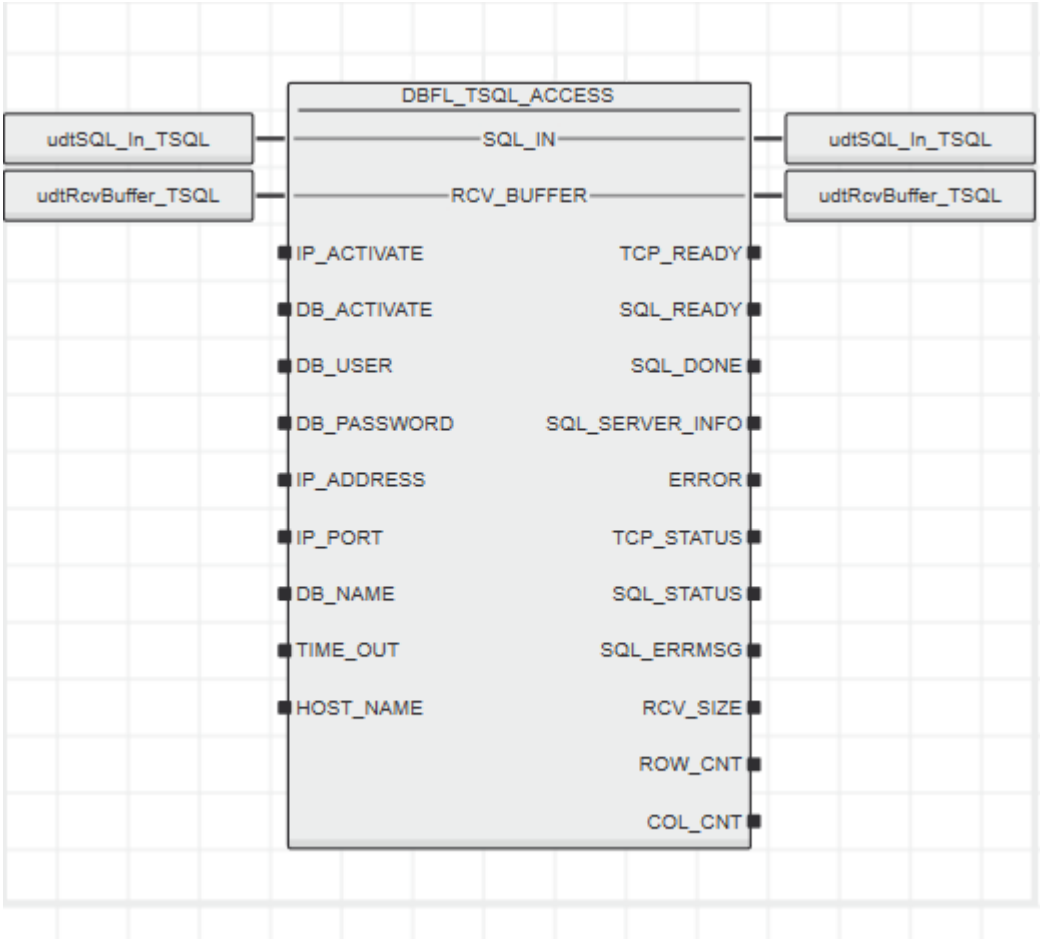
When specifying the IP_ADDRESS parameter, make sure that the syntax is correct: "172.16.252.10".

The DBFL_TSQL_ACCESS function block sends an SQL command to the database. The SQL command must be present at the SQL_IN input and must be an "ARRAY OF BYTE". To generate the SQL command, the SQL_CODE function block can be used, for example.

To terminate the SQL command, the DB_ACTIVATE input must be set to TRUE on connection establishment. Once the SQL command has been sent successfully, the SQL_READY output is set to TRUE.

The function block can only process one command. The maximum size of an SQL command is 1439 bytes.

5.1 Function block call



5.2 Input parameters

Name	Type	Description
IP_ACTIVATE	BOOL	TRUE: TCP connection to the database is established. FALSE: TCP connection is terminated.
DB_ACTIVATE	BOOL	Positive edge: SQL command is sent.
DB_USER	STRING	User name for the database connection
DB_PASSWORD	STRING	Password for the database connection
IP_ADDRESS	STRING	IP address of the database server
IP_PORT	STRING	Database server port (Default: MSSQL = 1433, if parameter is empty)
DB_NAME	STRING	Name of the database for the database connection
TIME_OUT	TIME	Timeout time for internal error routines (default = 5 s, if unwired, maximum 9999 s)
HOST_NAME	STRING	Hostname of the database connection Default: "PCWORX"

5.3 Output parameters

Name	Type	Description
TCP_READY	BOOL	TRUE: TCP connection to the database has been established. FALSE: TCP connection to the database has not been established.
SQL_READY	BOOL	SQL command has been sent.
SQL_DONE	BOOL	TRUE: SQL command has been executed and DB_ACTIVATE = TRUE.
SQL_SERVER_INFO	DBFL_UDT _MSSQL_S erver_Info	Provides the developer with status information about the SQL server. Structure of the DBFL_UDT_MSSQL_Server_Info data type (see Table 4-3).
ERROR	DINT	TCP error code (see Table "TCP error codes")
SQL_STATUS	DINT	SQL error code (see Table "MSSQL error codes")
SQL_ERRMSG	DBFL_UDT _STRING25 5	Corresponding error text for SQL_STATUS (direct from the SQL server)
RCV_SIZE	DINT	Size of the received user data, e.g., for a "SELECT * FROM MyTable". A maximum of 1439 bytes of user data can be received.
ROW_CNT	DINT	Number of affected rows (requires the SQL_DECODE function block for evaluation), must be connected to MAX_ROW when a table is received.
COL_CNT	DINT	Number of affected columns (requires the SQL_DECODE function block for evaluation), must be connected to MAX_COL when a table is received.

Explanation of the difference between SQL_DONE and SQL_READY

SQL_DONE is statically TRUE, SQL_READY generates a positive edge if DB_ACTIVATE = TRUE.

5.4 Inout parameters

Name	Type	Description
SQL_IN	DBFL_ARR _BYTE_0_1 439	SQL command in the form of an "ARRAY OF BYTE"
RCV_BUFFER	DBFL_ARR _BYTE_0_10219	Contains the data received from the server (can be evaluated with the SQL_DECODE function block).

5.5 TCP error codes

DiagCode	Description
16#0000	No errors have occurred
16#0001	Failed to create one or more tasks.
16#0002	Socket interface initialization failed (WinNT only).
16#0003	Dynamic memory could not be reserved.
16#0004	Function block cannot be initialized, as an error occurred when starting the asynchronous communication task.
16#0010	Socket initialization failed.
16#0011	Error sending a message
16#0012	Error receiving a message
16#0013	Unknown service code in the telegram header
16#0021	Invalid state transition on connection establishment
16#0030	No free channels available.
16#0031	Connection aborted.
16#0033	General timeout: Receiver not responding. Transmitter has not finished transmitting.
16#0034	Connection request was rejected.
16#0035	Receiver has not confirmed transmission, receiver may be overloaded (repeat transmission).
16#0040	Partner string is too long (255 characters, maximum).
16#0041	The specified IP address is invalid or could not be assigned.
16#0042	Invalid port number
16#0050	Transmission attempt on invalid connection (transmitter or receiver)
16#0053	All available connections are being used.
16#0061	No receiver confirmation. An invalid sequence number is being used.
16#0062	Transmitter and receiver data types do not match.
16#0063	Receiver is currently not ready to receive (possible causes: receiver is switched off or in the process of transferring data).
16#0064	A receive block with the corresponding R_ID cannot be found.
16#0065	Another function block instance is already operating on this connection.
16#0070	The partner controller is not configured as a time server.
16#FFFF	Internal socket error (connection to the database is reestablished automatically: duration 3 s)

5.6 MSSQL error codes

DiagCode	Description
16#22	Not an SQL command
16#255	The received data volume is too large. Table contains too many entries.
16#...	SQL error number Database login error (e.g., incorrect user name, invalid or incorrect password) or invalid SQL command (e.g., incorrect syntax). For more detailed information about SQL errors, please refer to the documentation of the database manufacturer.
16#33	Timeout SQL server does not respond within the specified “timeout” time.

5.7 Differences between MySQL and MSSQL error codes

In the event of errors, both libraries return corresponding error codes. There are considerably more error codes in MSSQL than in MySQL. In the event of an error, the server returns a descriptive error text. If this information is not sufficient enough to resolve the problem, please look up the description of the error codes on the MySQL and MSSQL websites.

MySQL error codes: http://dev.mysql.com/doc/refman/5.1/en/error-handling.html

(for client and server)

MSSQL error codes: http://www.microsoft.com

6 DBFL_MySQL_ACCESS

The function block enables access to a database that has been created on an MySQL / MariaDB server. The parameters required for the connection (DB_USER, DB_PASSWORD, IP_PORT, IP_ADDRESS, DB_NAME) must be a STRING.

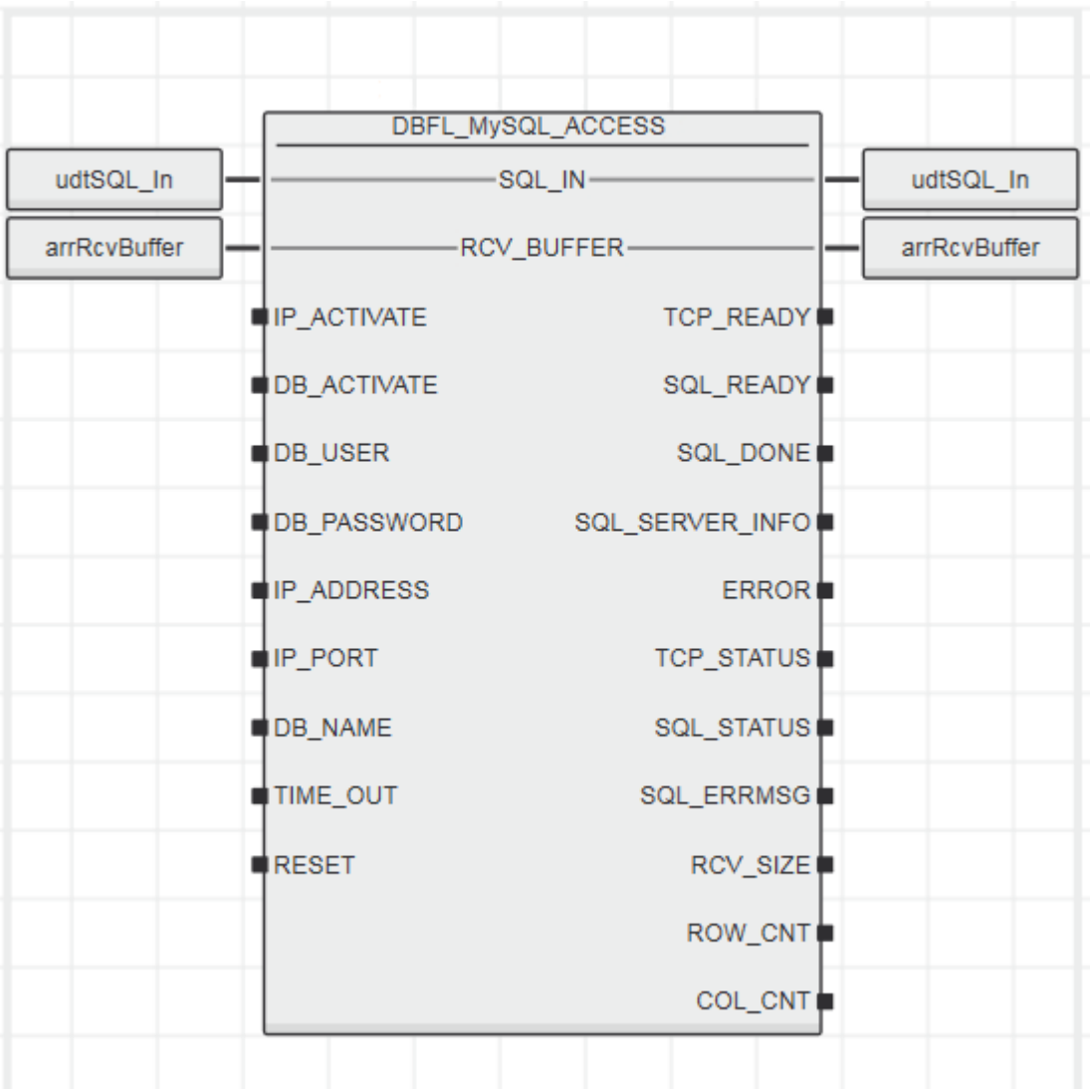
When specifying the IP_ADDRESS parameter, make sure that the syntax is correct: "172.16.252.10".

The DBFL_MySQL_ACCESS function block sends the SQL command to the database. The SQL command must be present at the SQL_IN input and must be an "ARRAY OF BYTE". To generate the SQL command, the SQL_CODE function block can be used, for example.

To send the SQL command, the DB_ACTIVATE input must be set to TRUE. Once the SQL command has been sent successfully, the SQL_READY output is set to TRUE.

The function block can only process one command. The maximum size of an SQL command is 1439 bytes.

6.1 Function block call



6.2 Input parameters

Name	Type	Description
IP_ACTIVATE	BOOL	TRUE: TCP connection to the database is established. FALSE: TCP connection is terminated.
DB_ACTIVATE	BOOL	Positive edge: SQL command is sent.
DB_USER	STRING	User name for the database connection
DB_PASSWORD	STRING	Password for the database connection
IP_ADDRESS	STRING	IP address of the database server
IP_PORT	STRING	Database server port (Default MySQL = 3306, if parameter is not set)
DB_NAME	STRING	Database name
TIME_OUT	TIME	Timeout time for internal error routines (default = 5 s, if unwired, maximum 9999 s)
RESET	BOOL	Reserved, only for internal diagnostics by the manufacturer.

6.3 Output parameters

Name	Type	Description
TCP_READY	BOOL	TRUE: TCP connection to the database has been established. FALSE: TCP connection to the database has not been established.
SQL_READY	BOOL	Function block is ready to send an SQL command.
SQL_DONE	BOOL	TRUE: SQL command has been executed successfully. The output is TRUE until the DB_ACTIVATE input is set to FALSE.
SQL_SERVER_INFO	DBFL_UDT_ MYSQL_ Server_Info	Provides the developer with status information about the SQL server. Structure of the DBFL_UDT_MYSQL_Server_Info data type (see Table).
ERROR	BOOL	TRUE: An error has occurred FALSE: No errors have occurred In the event of an error, the TCP_STATUS and SQL_STATUS outputs must be evaluated.
TCP_STATUS	DINT	TCP error code (see Table "TCP error codes")
SQL_STATUS	DINT	SQL error code (see http://dev.mysql.com/doc/refman/5.1/en/error-handling.html)
SQL_ERRMSG	DBFL_UDT_ STRING255	Corresponding error text for SQL_STATUS (direct from the SQL server)
RCV_SIZE	DINT	Size of the received user data, e.g., for a "select * from MyTable".
ROW_CNT	DINT	Number of affected rows (requires the SQL_DECODE function block for evaluation and must be connected to MAX_ROW when a table is received).
COL_CNT	DINT	Number of affected columns (requires the SQL_DECODE function block for evaluation, must be connected to MAX_COL when a table is received).

6.4 Inout parameters

Name	Type	Description
SQL_IN	DBFL_ARR *BYTE_0* 1439	SQL command in the form of an "ARRAY OF BYTE"
RCV_BUFFER	DBFL_ARR *BYTE_0* 1439	Contains the data received from the MySQL protocol (can be evaluated with the MYSQL_DECODE function block).

6.5 TCP error codes

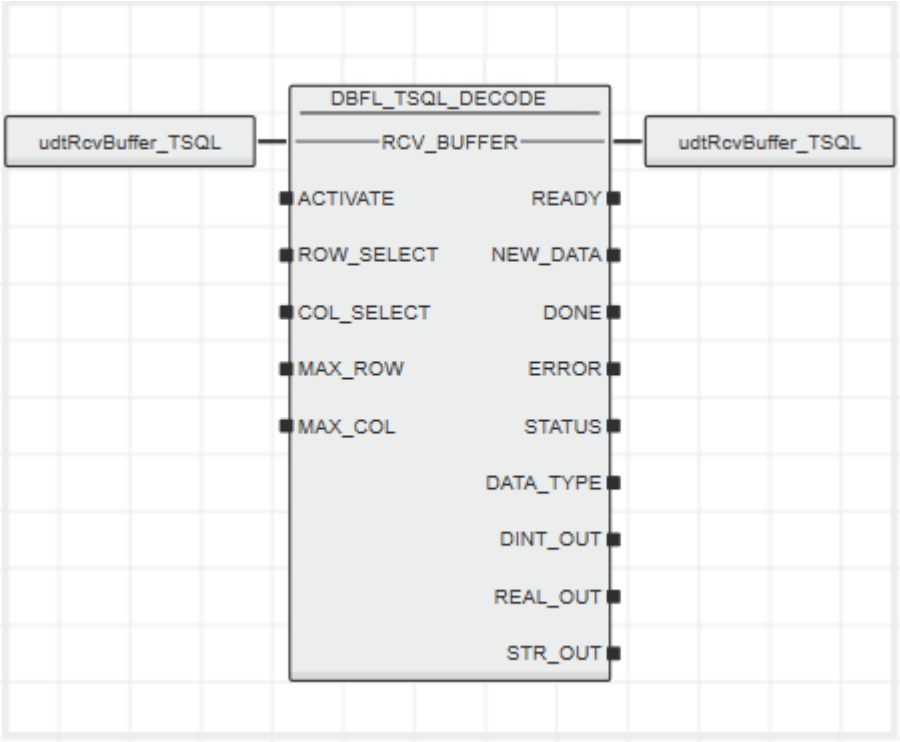
DiagCode	Description
20000	Internal socket error
20010	Socket error while sending data
20020	Internal socket error (connection to the database is reinitialized automatically: duration 3 s)
50000 + _IP_CONNECT:STAT US	Error codes (error code - 50000) of IP_CONNECT (see PC WorX online help)
60000 + _IP_USEND:STATUS	Error codes (error code - 60000) of IP_USEND (see PC WorX online help)
70000 + _IP_URCV:STATUS	Error codes (error code - 70000) of IP_URCV (see PC WorX online help)
79999	The request resulted in too much data in the receive buffer.

7 DBFL_TSQL_DECODE

The function block is used to evaluate a received table and is used as a continuation block of DBFL_TSQL_ACCESS.

The ROW_SELECT and COL_SELECT input parameters can be used to select a cell. The value TRUE at the ACTIVATE input triggers decoding of the selected cell. The DATA_TYPE output specifies the data type of the selected cell. Depending on the data type, the selected cell is output via STR_OUT, REAL_OUT, and DINT_OUT.

7.1 Function block call



7.2 Input parameters

Name	Type	Description
ACTIVATE	BOOL	Positive edge: Decoding of received data at the RCV_BUFFER input
ROW_SELECT	DINT	Specification of row for evaluation
COL_SELECT	DINT	Specification of column for evaluation
MAX_ROW	DINT	Maximum number of rows in the selected table (ROW_CNT of DBFL_TSQL_ACCESS function block)
MAX_COL	DINT	Maximum number of columns in the selected table (COL_CNT of DBFL_TSQL_ACCESS function block)

7.3 Output parameters

Name	Type	Description
READY	BOOL	TRUE: Data decoded successfully and ACTIVATE input = TRUE.
NEW_DATA	BOOL	Positive edge: New data is present at the corresponding output.
DONE	BOOL	TRUE: Command has been executed.
ERROR	BOOL	TRUE: An error has occurred FALSE: No errors have occurred
STATUS	INT	In the event of an error (ERROR = TRUE), the status can be evaluated.
DATA_TYPE	STRING	Data type of the selected cell
DINT_OUT	DINT	Contents of the selected cell (if INTEGER data type)
REAL_OUT	REAL	Contents of the selected cell (if REAL data type)
STR_OUT	STRING	Contents of the selected cell (if STRING data type)

7.4 Inout parameters

Name	Type	Description
RCV_BUFFER	DBFL_ARR *BYTE_0* 1439	Received data (from SQL_ACCESS function block)

7.5 Codes for STATUS output

Code	Description
0	No error
1	Column error (COL_SELECT > MAX_COL)
2	Row error (ROW_SELECT > MAX_ROW)
3	Invalid data type
4	Selected row/column is 0 (COL_SELECT or ROW_SELECT = 0)
5	Invalid data

7.6 MSSQL

MSSQL data type	Controller data type
INT	DINT formatted: Decimal
SMALLINT	DINT formatted: Decimal
TINYINT	DINT formatted: Decimal
BIGINT	STRING unformatted: Hexadecimal
CHAR(N)	STRING formatted: Alphanumeric
NCHAR(N)	STRING formatted: Alphanumeric
NVARCHAR(N)	STRING formatted: Alphanumeric
VARCHAR(N)	STRING formatted: Alphanumeric
REAL	REAL formatted: Decimal
FLOAT	STRING unformatted: Hexadecimal
NUMERIC(X,Y)	STRING unformatted: Hexadecimal
DECIMAL(X,Y)	STRING unformatted: Hexadecimal
DATETIME	STRING formatted: Decimal (for maximum resolution, see SQL server documentation)
SMALLDATETIME	STRING formatted: Decimal (for maximum resolution, see SQL server documentation)
TIMESTAMP	STRING unformatted: Hexadecimal
MONEY	STRING formatted: Decimal
SMALLMONEY	STRING formatted: Decimal
BIT	DINT formatted: Decimal
TEXT	Not supported
NTEXT	Not supported
UNIQUEIDENTIFIER	STRING formatted: Hexadecimal
BINARY(N)	STRING unformatted: Hexadecimal
VARBINARY(N)	STRING unformatted: Hexadecimal
IMAGE	Not supported
XML	Not supported

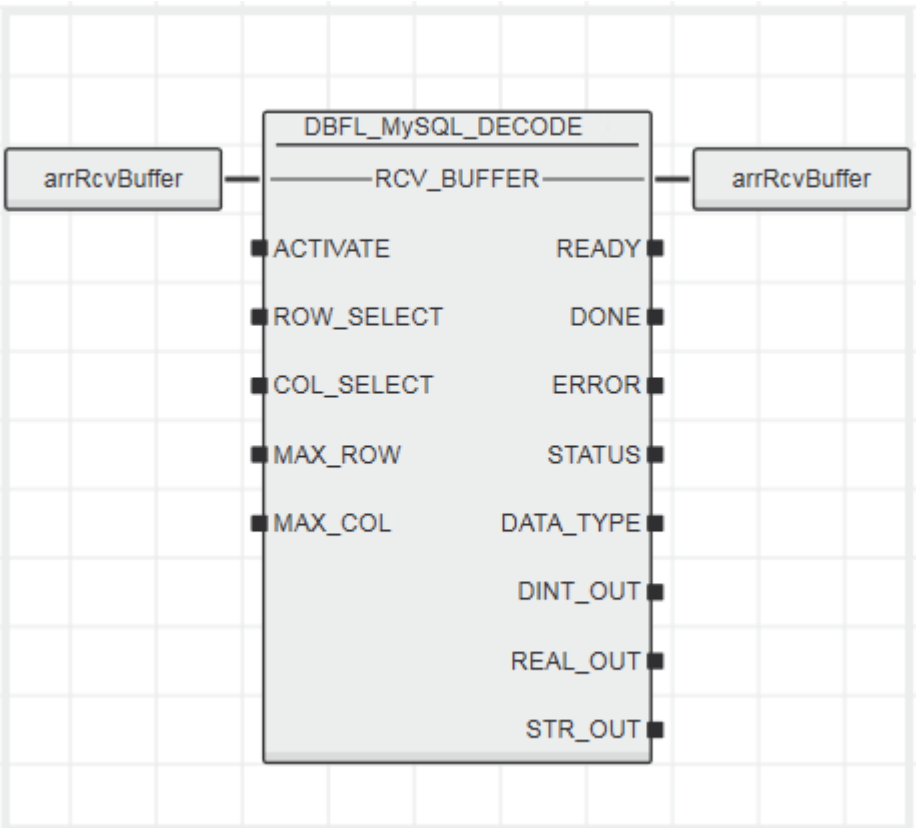
8 DBFL_MySQL_DECODE

The function block is used to evaluate a received table and can be used as a continuation block of DBFL_MySQL_ACCESS.

The value TRUE at the ACTIVATE input triggers the decoding of the selected cell. The ROW_SELECT and COL_SELECT input parameters can be used to select a cell. The DATA_TYPE output specifies the data type of the selected cell. Depending on the data type, the selected cell can be output via DINT_OUT, REAL_OUT, and STR_OUT.

Not all data types can be mapped to the controller. Please observe the tables for the used database. Convert the data types accordingly.

8.1 Function block call



8.2 Input parameters

Name	Type	Description
ACTIVATE	BOOL	Positive edge: Decoding of received data at the RCV_BUFFER input
ROW_SELECT	DINT	Specification of row for evaluation
COL_SELECT	DINT	Specification of column for evaluation
MAX_ROW	DINT	Maximum number of rows in the selected table (ROW_CNT of DBFL_MySQL_ACCESS function block)
MAX_COL	DINT	Maximum number of columns in the selected table (COL_CNT of DBFL_MySQL_ACCESS function block)

8.3 Output parameters

Name	Type	Description
READY	BOOL	TRUE: Data decoded successfully
DONE	BOOL	TRUE: Command has been executed.
ERROR	BOOL	TRUE: An error has occurred FALSE: No errors have occurred
STATUS	INT	In the event of an error (ERROR = TRUE), the status can be evaluated.
DATA_TYPE	STRING	Data type of the selected cell
DINT_OUT	DINT	Contents of the selected cell (if INTEGER data type)
REAL_OUT	REAL	Contents of the selected cell (if REAL data type)
STR_OUT	STRING	Contents of the selected cell (if STRING data type)

8.4 Inout parameters

Name	Type	Description
RCV_BUFFER	DBFL_ARR *BYTE_0* 1439	Received data (from MySQL_ACCESS function block)

8.5 Codes for status output

Code	Description
0	No error
1	Column error (COL_SELECT > MAX_COL)
2	Row error (ROW_SELECT > MAX_ROW)
3	Invalid data type
4	Selected row/column is 0 (COL_SELECT or ROW_SELECT = 0)
5	Invalid data

8.6 MySQL

MySQL data type	Controller data type
BIT	STRING formatted: Hexadecimal
TINYINT	DINT formatted: Decimal
SMALLINT	DINT formatted: Decimal
MEDIUMINT	DINT formatted: Decimal
INT	DINT formatted: Decimal
INTEGER	DINT formatted: Decimal
BIGINT	DINT formatted: Decimal
FLOAT	REAL formatted: Decimal
DOUBLE	REAL formatted: Decimal
DOUBLE PRECISION	REAL formatted: Decimal
REAL	REAL formatted: Decimal
DECIMAL	REAL formatted: Decimal
DEC	REAL formatted: Decimal
DATE	STRING formatted: Alphanumeric
DATETIME	STRING formatted: Alphanumeric
TIMESTAMP	STRING formatted: Alphanumeric
TIME	STRING formatted: Alphanumeric
YEAR	STRING formatted: Alphanumeric
CHAR	STRING formatted: Alphanumeric
TINYBLOB	STRING formatted: Hexadecimal
TINYTEXT	STRING formatted: Alphanumeric
BLOB	STRING formatted: Hexadecimal
TEXT	STRING formatted: Alphanumeric
MEDIUMBLOB	STRING formatted: Hexadecimal
MEDIUMTEXT	STRING formatted: Alphanumeric
LONGBLOB	STRING formatted: Hexadecimal
LONGTEXT	STRING formatted: Alphanumeric
ENUM	STRING formatted: Alphanumeric
SET	STRING formatted: Alphanumeric

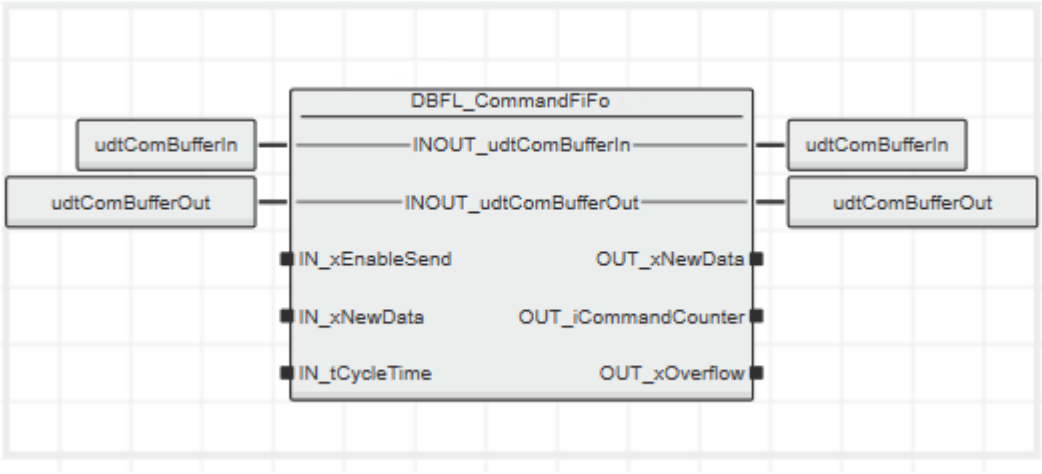
"Unsigned" data types (unsigned flag is set) are output as formatted decimal strings in the controller. (Var) binary data types (binary flag is set) are output as formatted hexadecimal strings in the controller. These also include the TEXT data type. The results of a saved procedure/function are often output as formatted hexadecimal strings in the controller. Hexadecimal strings can be processed as arithmetic operations.

9 DBFL_CommandFiFo

The function block saves up to 50 SQL commands.

FIFO (first in/first out) means that commands are called in the same order they are processed.

9.1 Function block call



9.2 Input parameters

Name	Type	Description
IN_xEnableSend	BOOL	TRUE: Enable sending. Telegram can be transmitted from the buffer to the transmit block.
IN_xNew Data	BOOL	Positive edge: Data from INOUT_udtComBufferIn is transferred to the buffer memory.
IN_tCycleTime	TIME	In case of TRUE at IN_xEnableSend this cycletime determines the sending out interval od commands to the send block

9.3 Output parameters

Name	Type	Description
OUT_xNewData	BOOL	TRUE: Entry of new data from the intermediate buffer in INOUT_udtComBufferOut
OUT_iCommandCou nter	INT	Number of saved telegrams
OUT_XOverflow	BOOL	TRUE: The intermediate buffer is full.

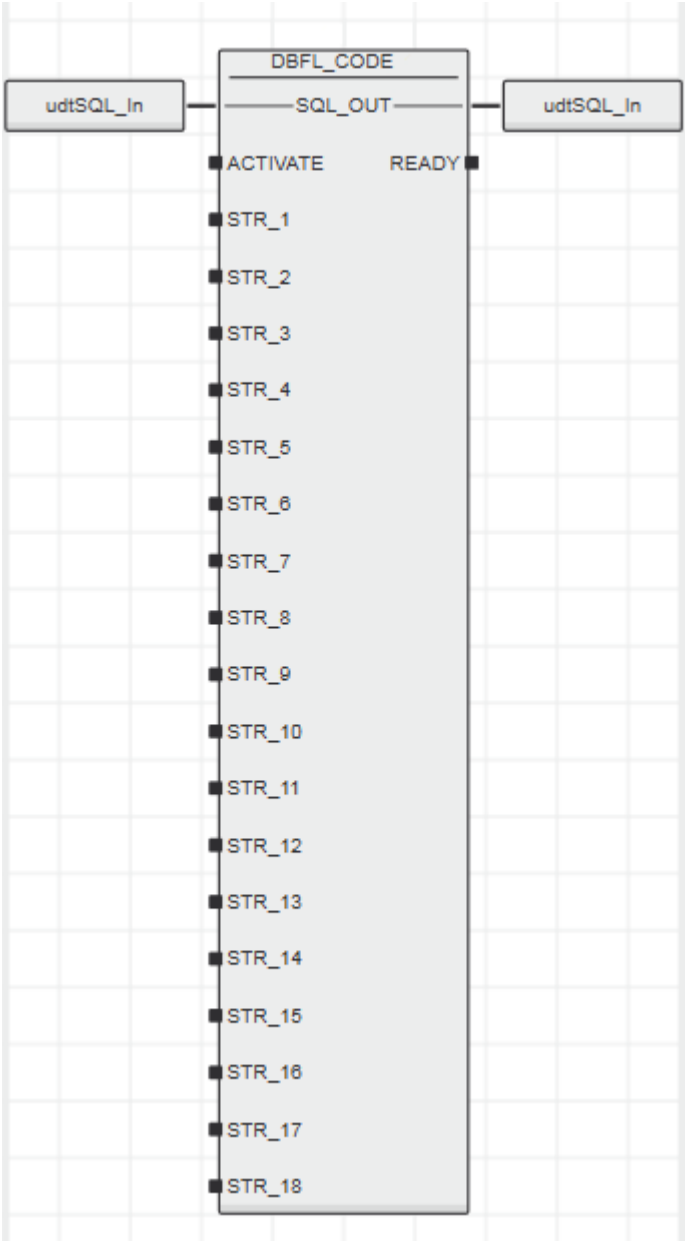
9.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer In	DBFL_UDT_SQL_COMMAND	SQL command input
INOUT_udtComBuffer Out	DBFL_UDT_SQL_COMMAND	SQL command output

10 DBFL_CODE

The function block inserts SQL commands or parts of them in the "SQL_OUT" array.

10.1 Function block call



10.2 Input parameters

Name	Type	Description
ACTIVATE	BOOL	Positive edge: The data present at inputs STR_1 to STR_18 is accepted by SQL_OUT.
STR_1 ... STR_18	STRING	Input string for database command (maximum of 79 characters per string supported)

10.3 Output parameters

Name	Type	Description
Ready	BOOL	TRUE: Data is accepted.

10.4 Inout parameters

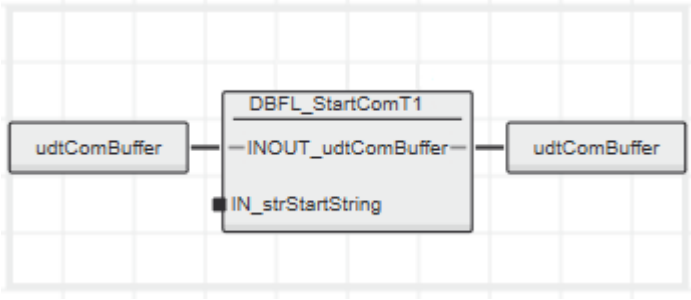
Name	Type	Description
SQL_OUT	DBFL_ARR_B YTE_0_1439	SQL command as byte array

11 DBFL_StartComT1

The function block creates the start of a database command. It deletes the SQL command in the buffer and inserts the first (new) command string that is present at the IN_strStartString input.

The following function blocks can be used as an alternative to the DBFL_CODE function block.

11.1 Function block call



11.2 Input parameters

Name	Type	Description
IN_strStartString	STRING	First part of the command that is transferred to the database.

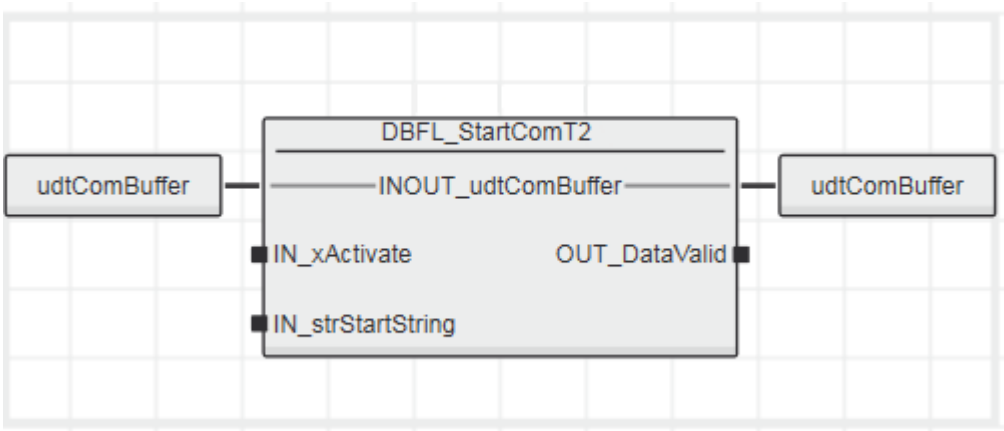
11.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

12 DBFL_StartComT2

The function block creates the start of a database command. It deletes the SQL command in the buffer and inserts the first command string.

12.1 Function block call



12.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: Value from IN_strStartString is entered in INOUT_udtComBuffer.
IN_strStartString	STRING	First part of the command that is transferred to the database.

12.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

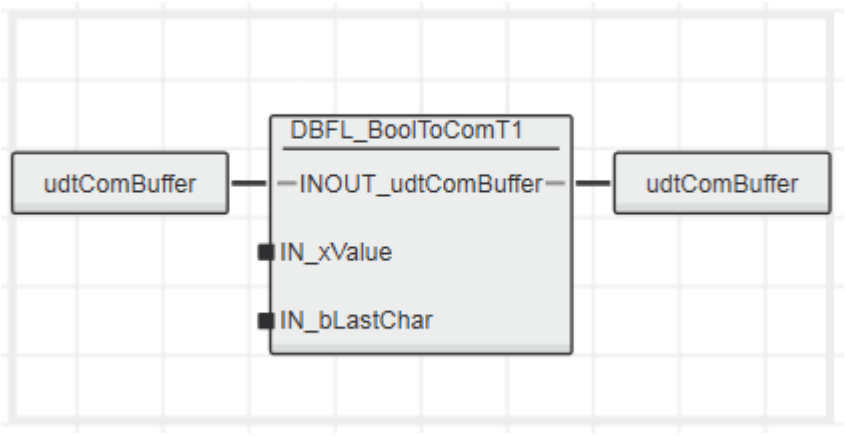
12.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

13 DBFL_BoolToComT1

The function block inserts a Boolean value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

13.1 Function block call



13.2 Input parameters

Name	Type	Description
IN_xValue	BOOL	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

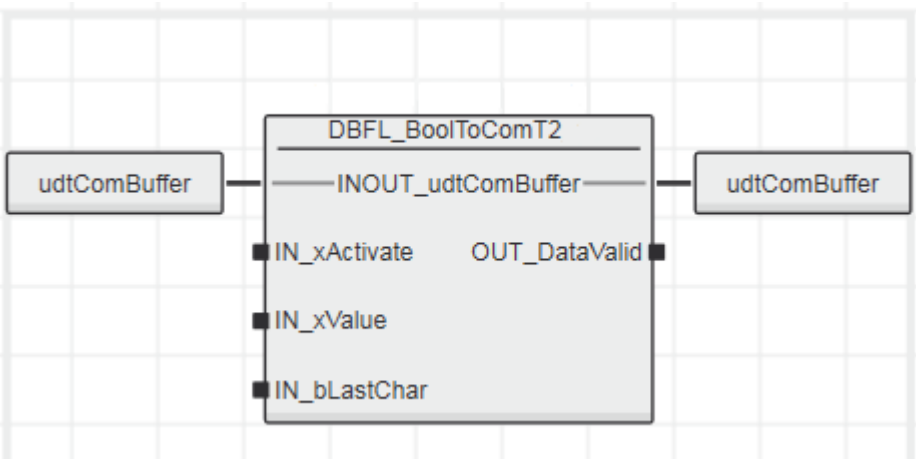
13.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

14 DBFL_BoolToComT2

The function block inserts a Boolean value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

14.1 Function block call



14.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: The value from IN_xValue is entered in INOUT_udtComBuffer.
IN_xValue	BOOL	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

14.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

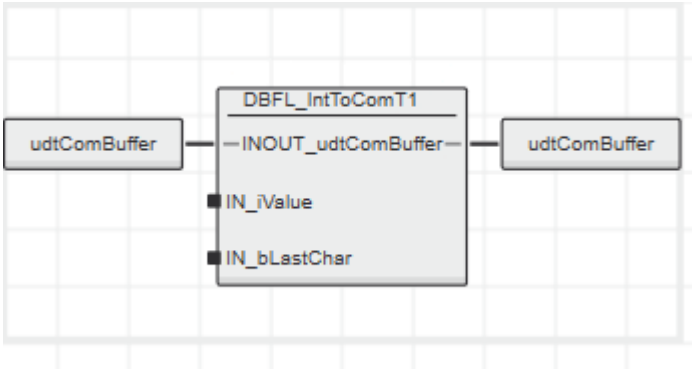
14.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

15 DBFL_IntToComT1

The function block inserts an integer value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

15.1 Function block call



15.2 Input parameters

Name	Type	Description
IN_iValue	INT	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

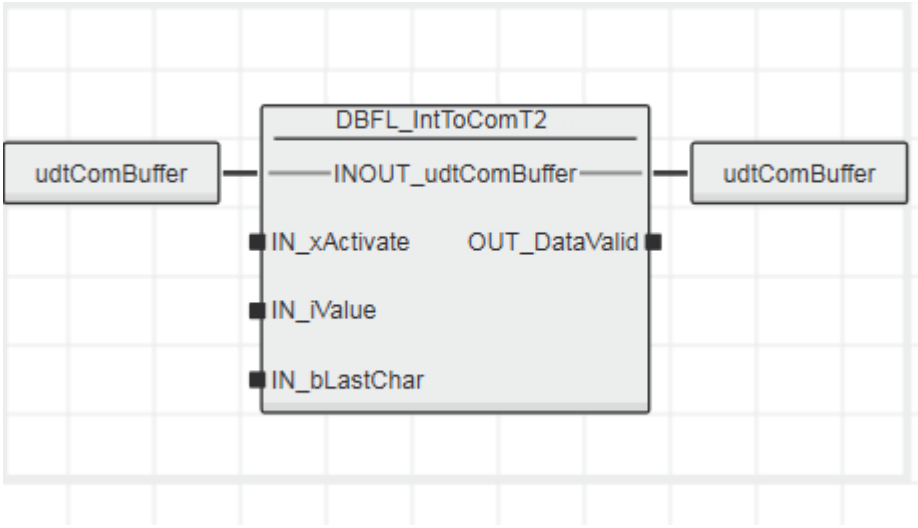
15.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

16 DBFL_IntToComT2

The function block inserts an integer value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

16.1 Function block call



16.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: The value from IN_iValue is entered in INOUT_udtComBuffer.
IN_iValue	INT	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

16.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

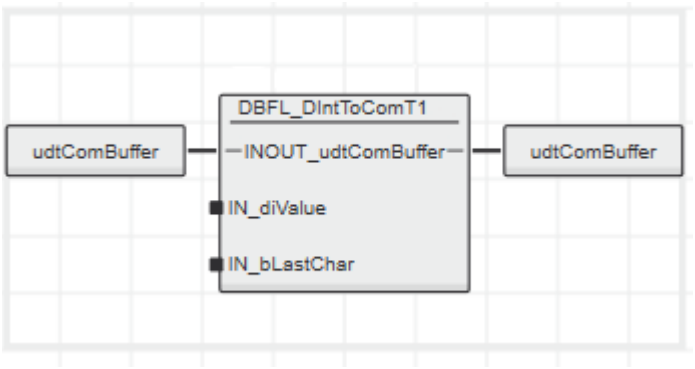
16.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

17 DBFL_DIntToComT1

The function block inserts a DINT value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

17.1 Function block call



17.2 Input parameters

Name	Type	Description
IN_diValue	DINT	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

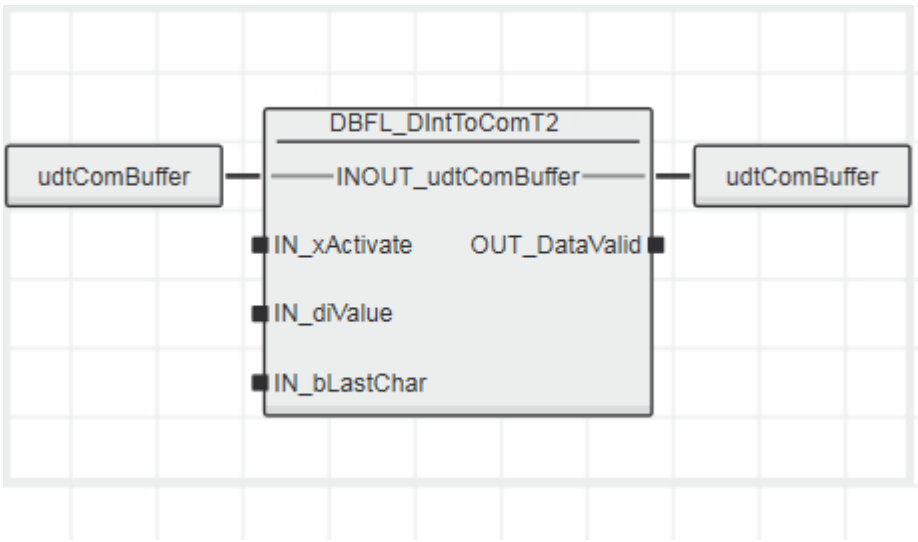
17.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

18 DBFL_DIntToComT2

The function block inserts a DINT value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

18.1 Function block call



18.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: Value from IN_diValue is entered in INOUT_udtComBuffer.
IN_diValue	DINT	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

18.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

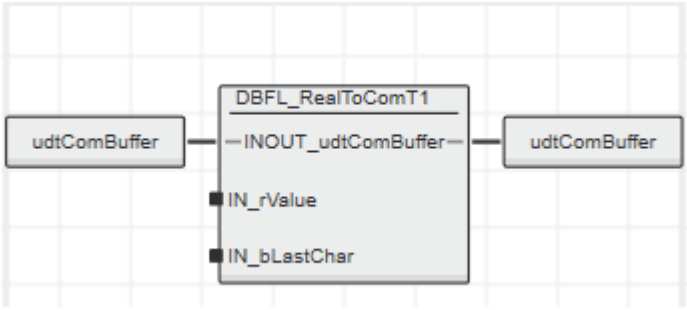
18.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

19 DBFL_RealToComT1

The function block inserts a REAL value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

19.1 Function block call



19.2 Input parameters

Name	Type	Description
IN_rValue	REAL	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

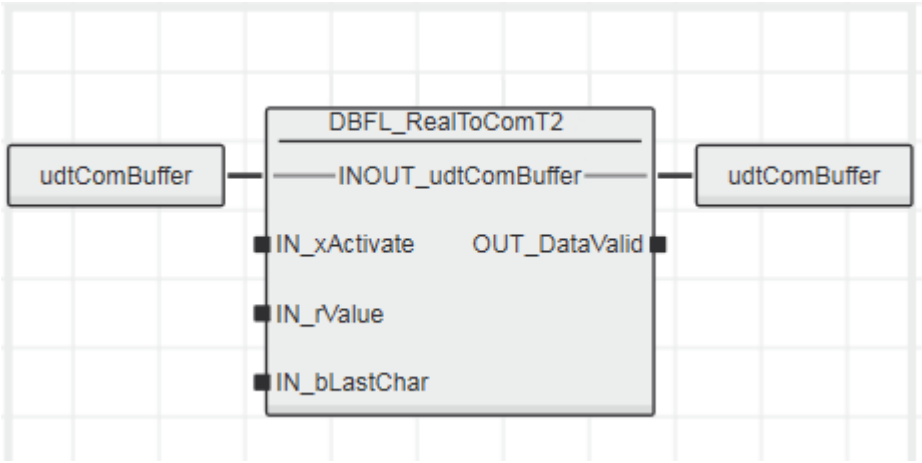
19.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

20 DBFL_RealToComT2

The function block inserts a REAL value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

20.1 Function block call



20.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: Value from IN_rValue is entered in INOUT_udtComBuffer.
IN_rValue	REAL	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

20.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

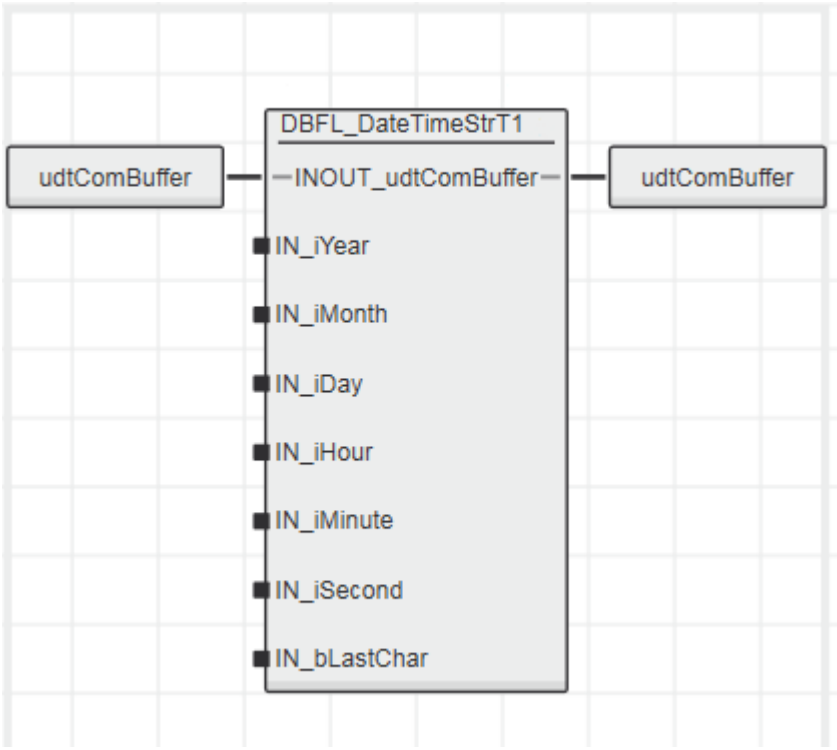
20.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

21 DBFL_DateTimeStrT1

The function block inserts a date/time value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

21.1 Function block call



21.2 Input parameters

Name	Type	Description
IN_iYear	INT	Date: Year
IN_iMonth	INT	Date: Month
IN_iDay	INT	Date: Day
IN_iHour	INT	Time: Hour
IN_iMinute	INT	Time: Minute
IN_iSecond	INT	Time: Second
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

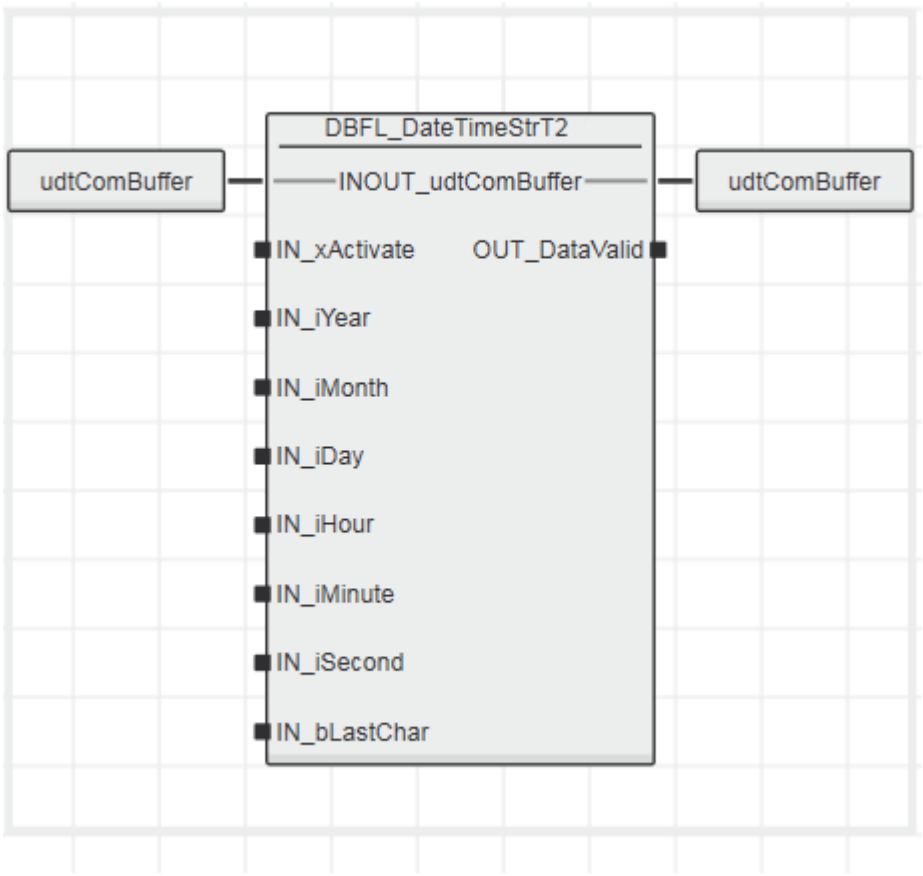
21.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

22 DBFL_DateTimeStrT2

The function block inserts a date/time value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

22.1 Function block call



22.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: The values from IN_iYear, IN_iMonth, IN_iDay, IN_iHour, IN_iMinute, IN_iSecond are entered in INOUT_udtComBuffer.
IN_iYear	INT	Date: Year
IN_iMonth	INT	Date: Month
IN_iDay	INT	Date: Day
IN_iHour	INT	Time: Hour
IN_iMinute	INT	Time: Minute
IN_iSecond	INT	Time: Second
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

22.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

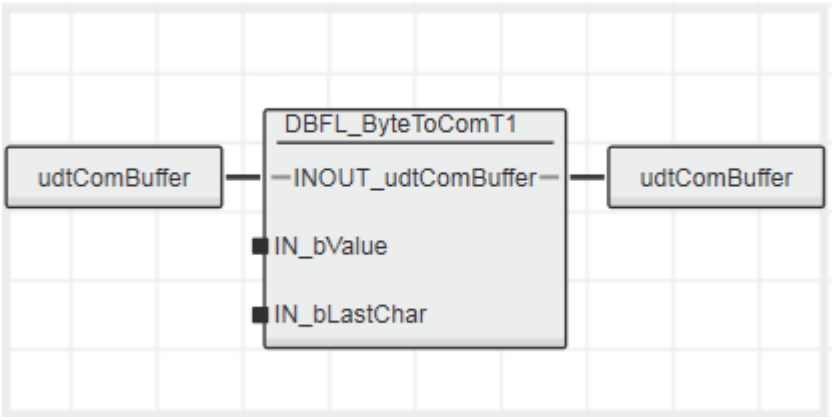
22.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

23 DBFL_ByteToComT1

The function block inserts a byte value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

23.1 Function block call



23.2 Input parameters

Name	Type	Description
IN_bValue	BYTE	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

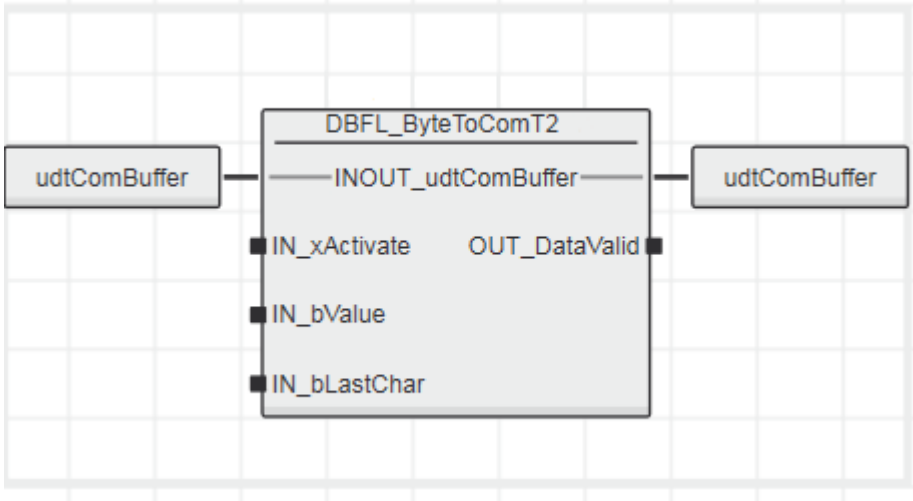
23.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

24 DBFL_ByteToComT2

The function block inserts a byte value in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

24.1 Function block call



24.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: Value from IN_bValue is entered in INOUT_udtComBuffer.
IN_bValue	BYTE	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

24.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

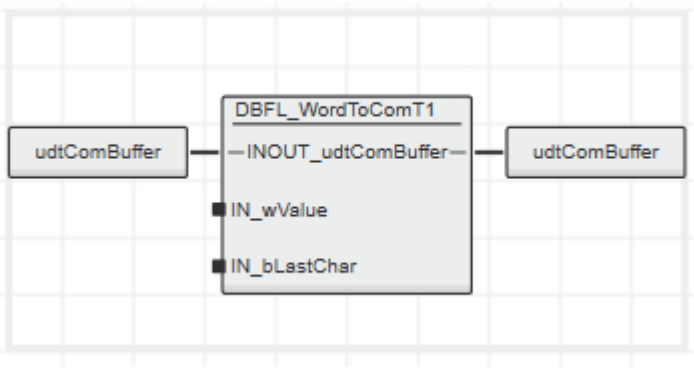
24.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

25 DBFL_WordToComT1

The function block inserts a data word in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

25.1 Function block call



25.2 Input parameters

Name	Type	Description
IN_wValue	WORD	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

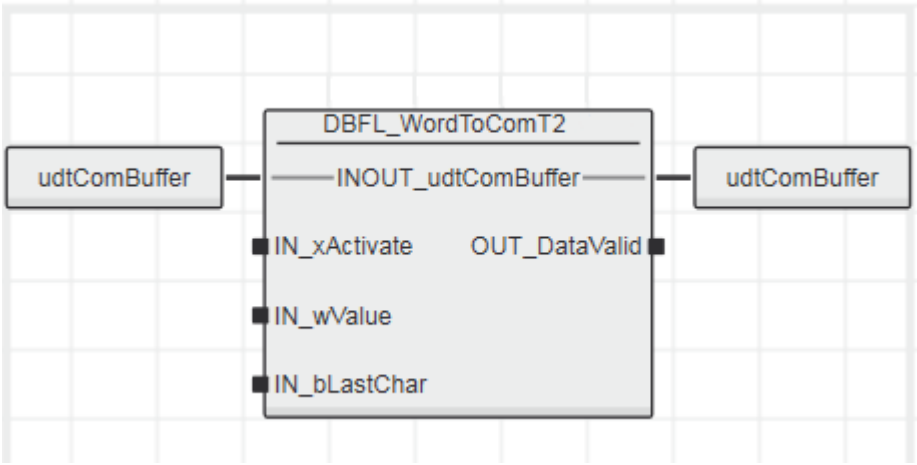
25.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

26 DBFL_WordToComT2

The function block inserts a data word in the SQL command. The IN_bLastChar input can be used to enter separators between values or the concluding character in the SQL command.

26.1 Function block call



26.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: The value from IN_wValue is entered in INOUT_udtComBuffer.
IN_wValue	WORD	Value that is transferred to the database.
IN_bLastChar	BYTE	Entry of separators between values or the concluding character: • ',' = BYTE#44 separator • ')' = BYTE#41 concluding character

26.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

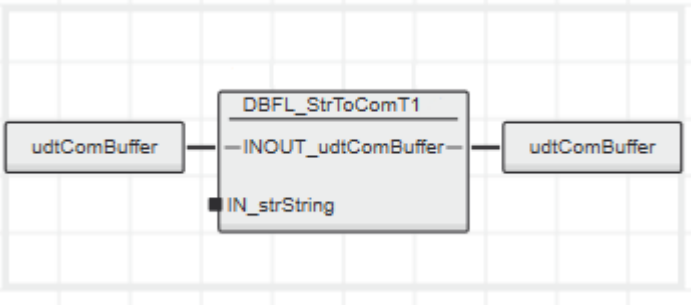
26.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

27 DBFL_StrToComT1

The function block inserts a string in the SQL command.

27.1 Function block call



27.2 Input parameters

Name	Type	Description
IN_strString	STRING	Value that is transferred to the database.

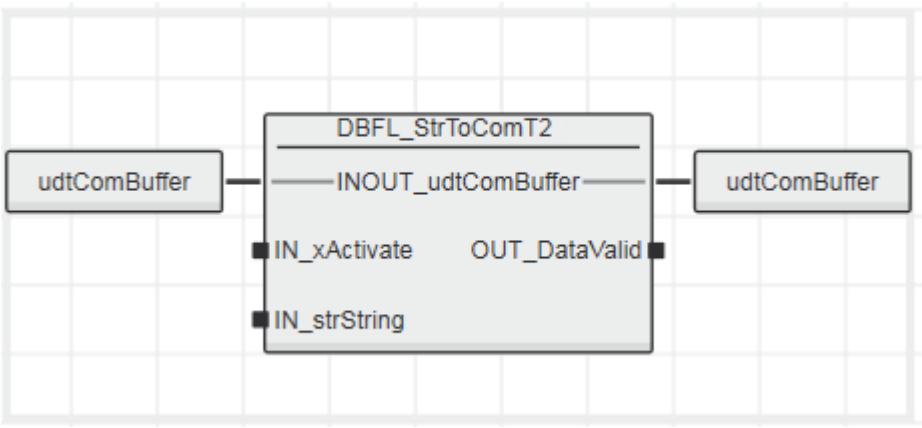
27.3 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

28 DBFL_StrToComT2

The function block inserts a string in the SQL command.

28.1 Function block call



28.2 Input parameters

Name	Type	Description
IN_xActivate	BOOL	Positive edge: Value from IN_strString is entered in INOUT_udtComBuffer.
IN_strString	STRING	Value that is transferred to the database.

28.3 Output parameters

Name	Type	Description
OUT_DataValid	BOOL	Positive edge: Start STRING has been entered in the IN/OUT buffer.

28.4 Inout parameters

Name	Type	Description
INOUT_udtComBuffer	DBFL_UDT_SQL_COMMAND	SQL command

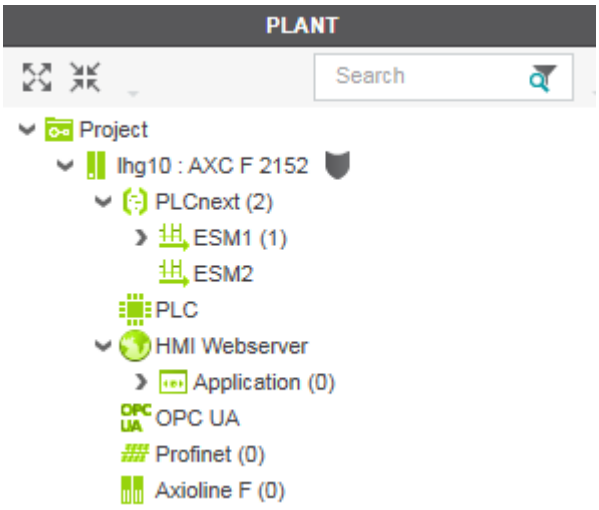
29 Startup examples

For the startup instruction please find the following examples:

- DBFL_SQL_5_EXA_MsSQL_AXC_F_2152.pcwex
- DBFL_SQL_5_EXA_MySQL_AXC_F_2152.pcwex

These examples are located in the “Examples” folder of the unzipped msi file of the library.

Plant:

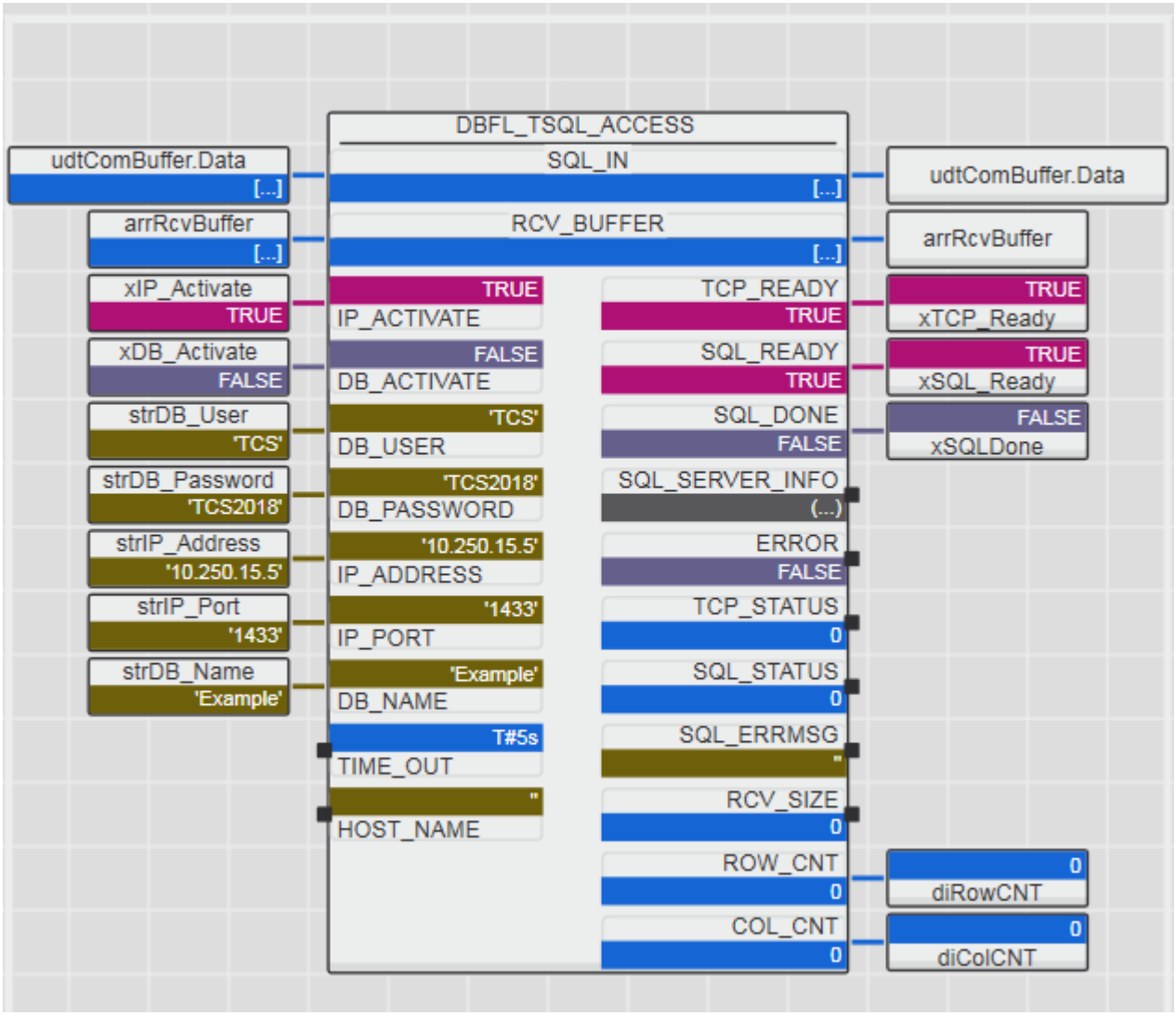


29.1 Example MsSQL

This section describes the use of the DBFL_SQL with the AXC F 2152 (2404267) and a MsSQL database.

29.1.1 Connect to database

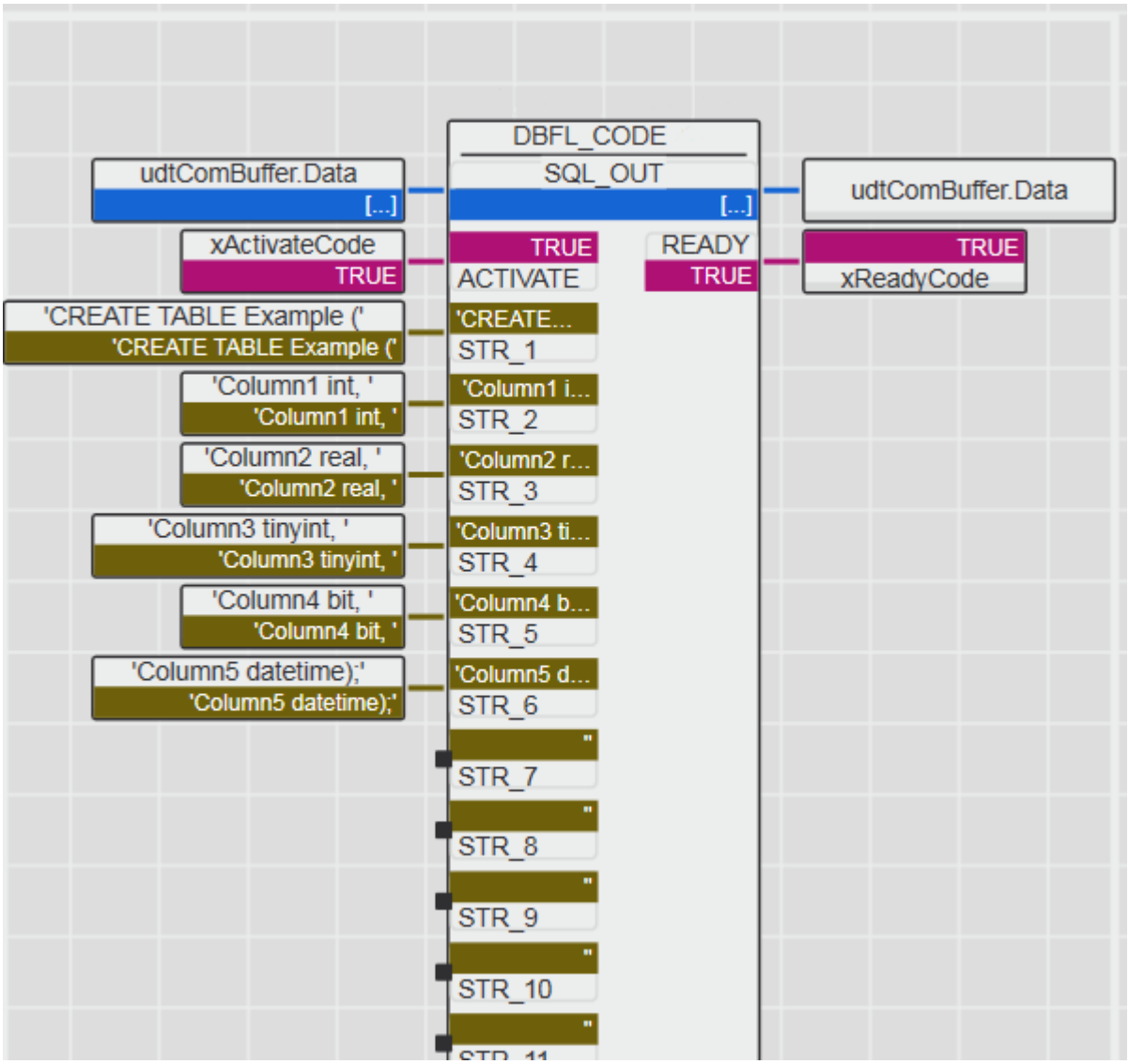
Initialize the variables of the DBFL_TSQL_ACCESS function block:



- Set the initialization value of the "strDB_User", "strDB_Password", "strIP_Adress", "strIP_Port" and "strDB_Name" variables with the Data from your database.
- Activate the variable "xIP_Activate" of the DBFL_TSQL_ACCESS function block by double-clicking and overwrite. An attempt is made to establish a connection to the database.
- TCP_Ready and SQL_Ready are set to TRUE.

29.1.2 Creating a database table

Initialize the variables of the DBFL_CODE function block:



- Set the initialization value of the "STR_1" variable with the SQL command “CREATE TABLE Example (".
- Set the initialization value of the "STR_2" variable with the SQL command "Column1 int,".
- Set the initialization value of the "STR_3" variable with the SQL command "Column2 real,".
- Set the initialization value of the "STR_4" variable with the SQL command "Column3 tinyint,".
- Set the initialization value of the "STR_5" variable with the SQL command "Column4 bit,".
- Set the initialization value of the "STR_6" variable with the SQL command "Column5 datetime);".

Activate the variable "xActivateCode" of the DBFL_CODE function block by double-clicking and overwrite. For this action the outputs TCP_Ready and SQL_Ready of the DBFL_TSQL_ACCESS function block have to be TRUE.

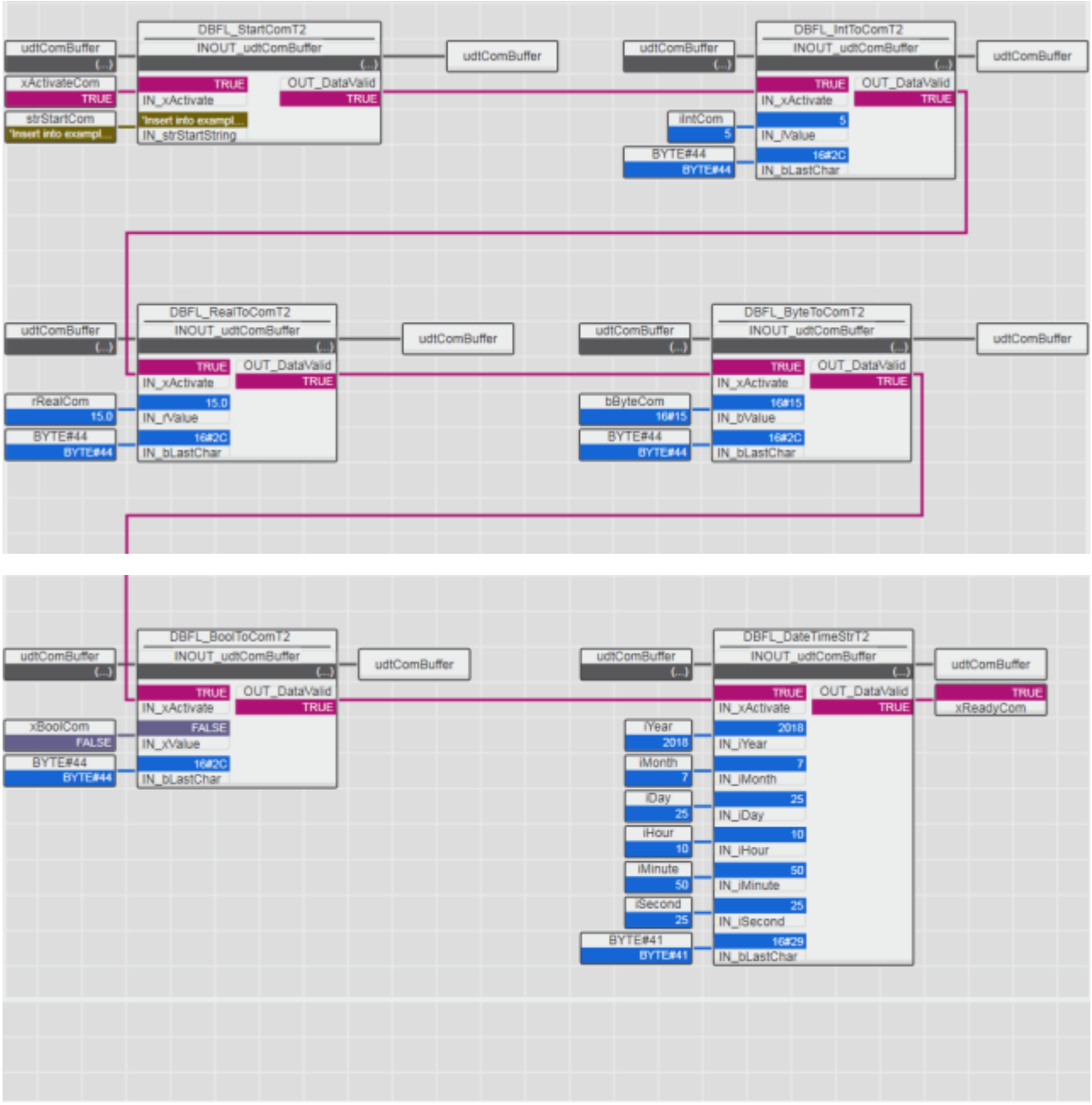
The SQL command is written to the udtComBuffer.Data and passed to the DBFL_TSQL_ACCESS function block. IF the xReadyCode output is TRUE, the xDB_Activate input of the DBFL_TSQL_ACCESS function block can be set to TRUE. If the transmission attempt is successful, the SQL_DONE output is set to TRUE at the DBFL_TSQL_Access function block.

29.1.3 Writing to the database

Initialize the variable "strStartCom" of the DBFL_StartComT2 function block with the SQL command "INSERT INTO Example VALUES (".

Initialize the value variables of the function blocks below:

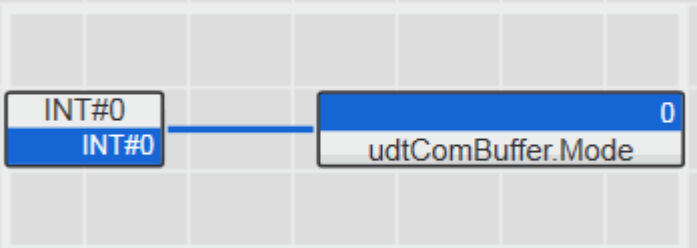
- DBFL_IntToComT2: The block inserts a value of type INT in the SQL command.
- DBFL_RealToComT2 The block inserts a value of type REAL in the SQL command.
- DBFL_ByteToComT2 The block inserts a value of type BYTE in the SQL command.
- DBFL_BoolToComT2 The block inserts a value of type BOOL in the SQL command.
- DBFL_DateTimeStrT2 The block inserts a date / time value in the SQL command.



Attention:

To generate a MsSQL compliant command, the udtComBuffer.Mode parameter must be assigned with other value than INT#1

Example: MsSqlCommand.Mode: = INT#0;



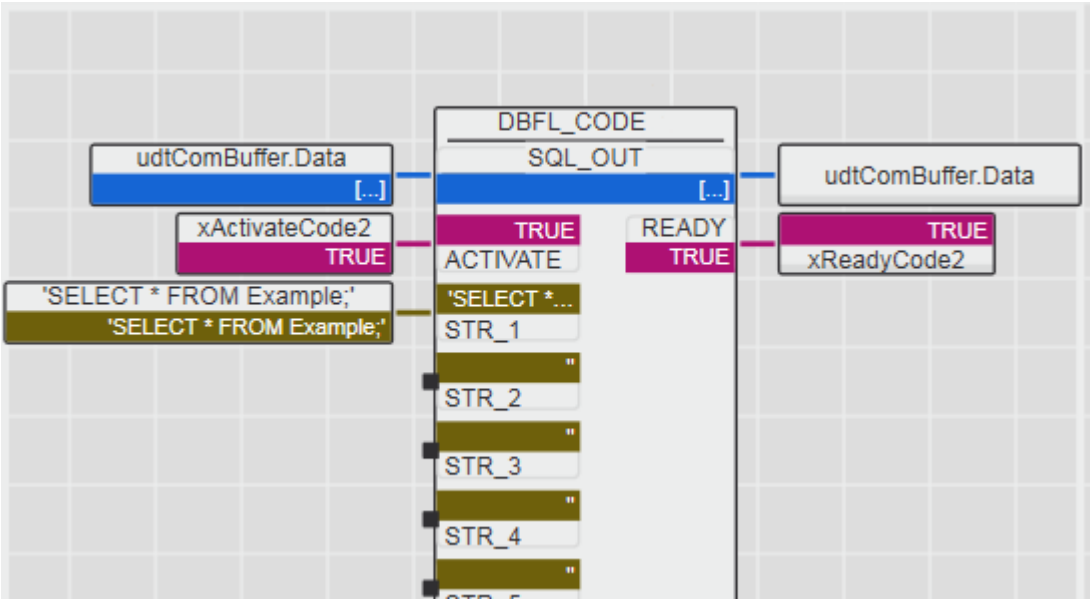
The IN_bLastChar input allows to enter the separator and terminator between the individual values in the SQL command. BYTE#44 = , BYTE#41 =)

The SQL command is written to the udtComBuffer and passed to the DBFL_TSQL_ACCESS function block. Make sure that the variables are written into the command in the correct order. If the xReadyCode output of the last *ToCom block is TRUE, set the
xDB_Activate input of the DBFL_TSQL_ACCESS function block also to TRUE. If the transmission attempt is successful, the SQL_DONE output is set to TRUE at the DBFL_TSQL_Access function block.

29.1.4 Reading data from a table

To read data from database initialize the variable "STR_1" of the DBFL_CODE function block with the SQL command "SELECT * FROM Example". With this command you read all data of the database. If you only want to read certain columns, specify them in the command. For example: 'select column1, column3 from example'

Activate the variable "xActivateCode2" of the DBFL_CODE function block by double-clicking and overwrite.



The SQL command is written to the udtComBuffer and passed to the DBFL_TSQL_ACCESS function block. If the transmission attempt is successful, the SQL_DONE output is set to TRUE at the DBFL_TSQL_Access function block.

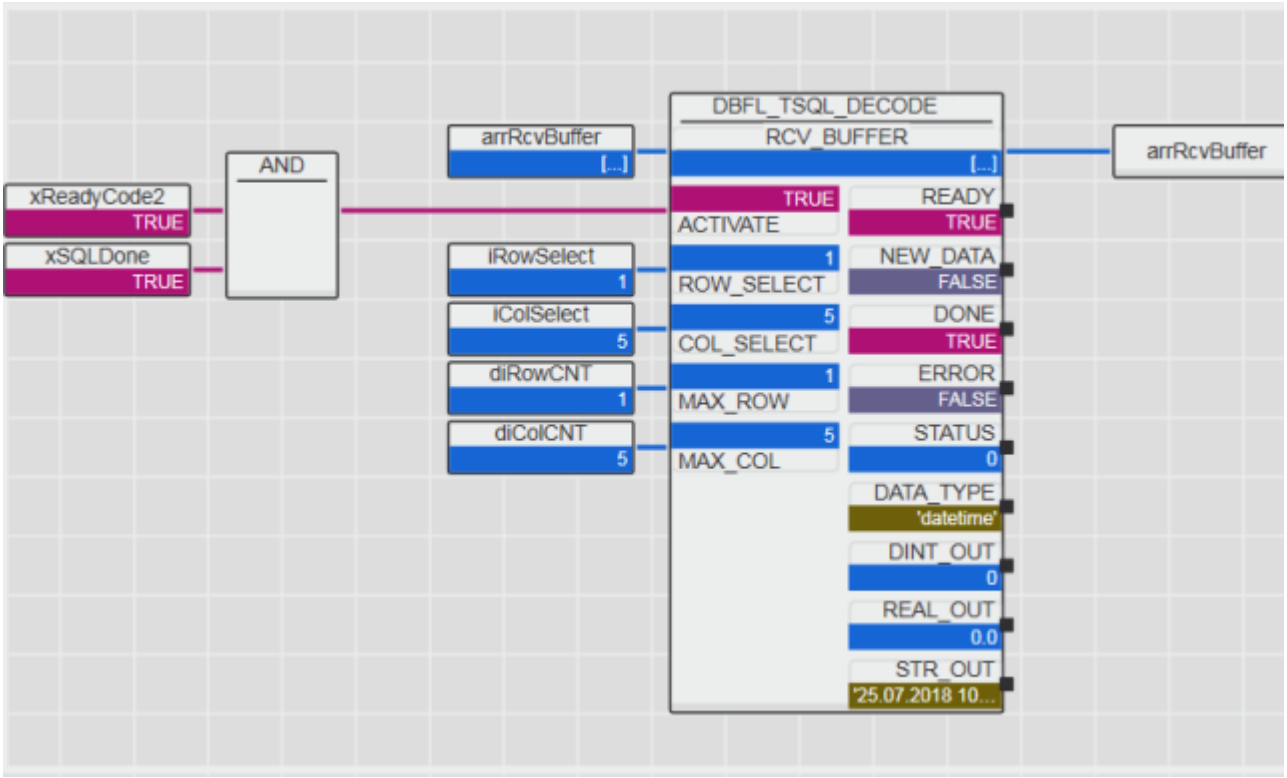
The data record returned by the database is decoded by the DBFL_TSQL_DECODE function block.

To select a row, set the "iRowSelect" parameter in the DBFL_TSQL_DECODE function block.

To select a column, set the "iColSelect" parameter in the DBFL_TSQL_DECODE function block.

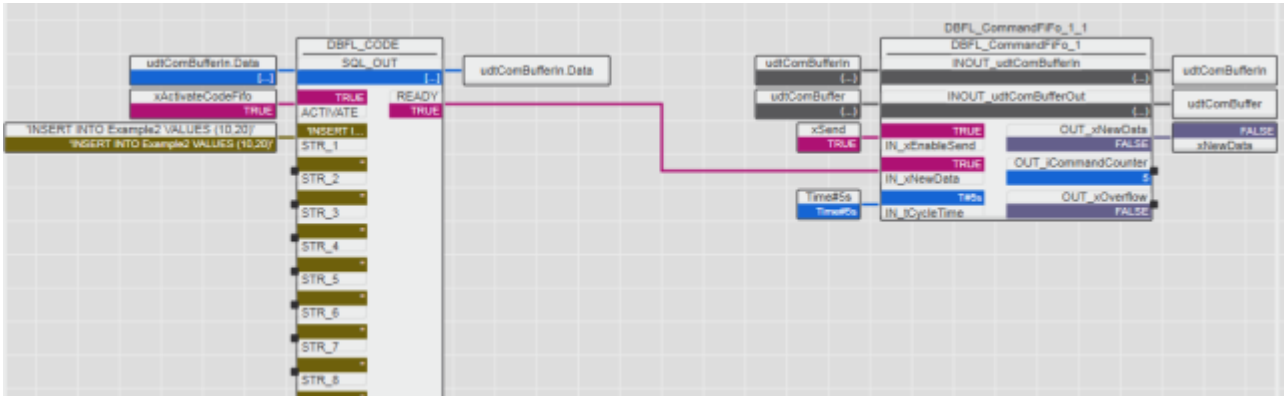
Activate the "xDecodeActivate" variable of the DBFL_DECODE function block by double-clicking and overwrite.

The data for the returned cell is now available at the corresponding output. In the example, the data type of the cell is datetime. The content is displayed at STR_OUT output.



29.1.5 CommandFiFo

An other option to send commands to the SQL database is the DBFL_CommandFiFo function block. With this function block you can save the commands and send them to the database (**F**irst **i**n **F**irst **o**ut)



For Example:

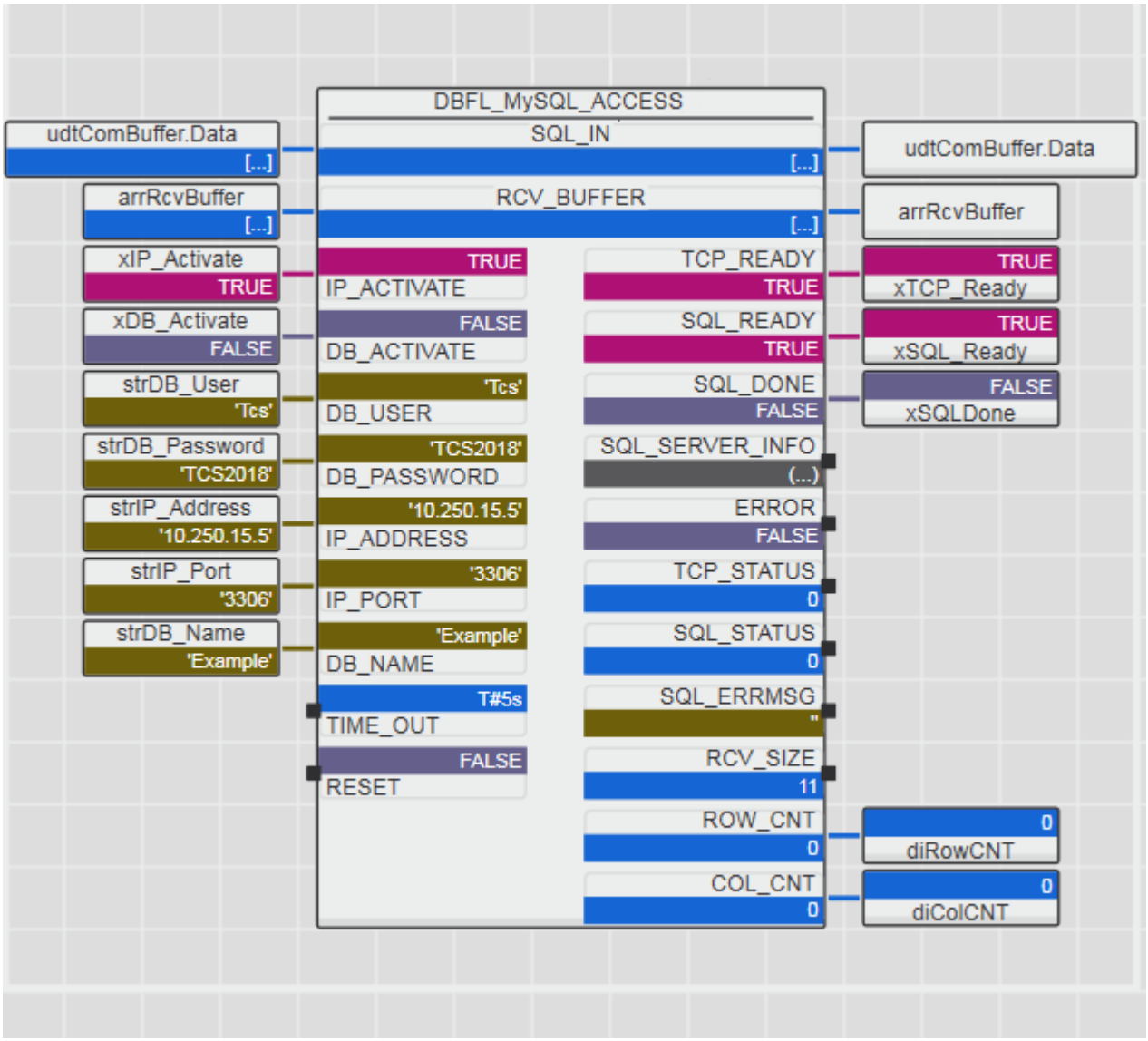
1. Activate the DBFL_Code function block
2. When the xReady of the DBFL_Code is True the xNewData of the CommandFiFo is set to TRUE
3. The command will be save in the buffer (iCommandCounter = 1)
4. Repeat step 1-3 a few times (iCommandCounter = x)
5. Activate the xEndableSend of the CommandFiFo
6. The commands are sent to the database according to the first in first out principle
7. With each new command, the output xNewData is set to True for one cycle
8. Connect output xNewData with the xDB_Activate input of the Access function block

29.2 Example MySQL

This section describes the use of the DBFL_SQL function block with the AXC F 1050 (2404701) and a MySQL database.

29.2.1 Connect to database

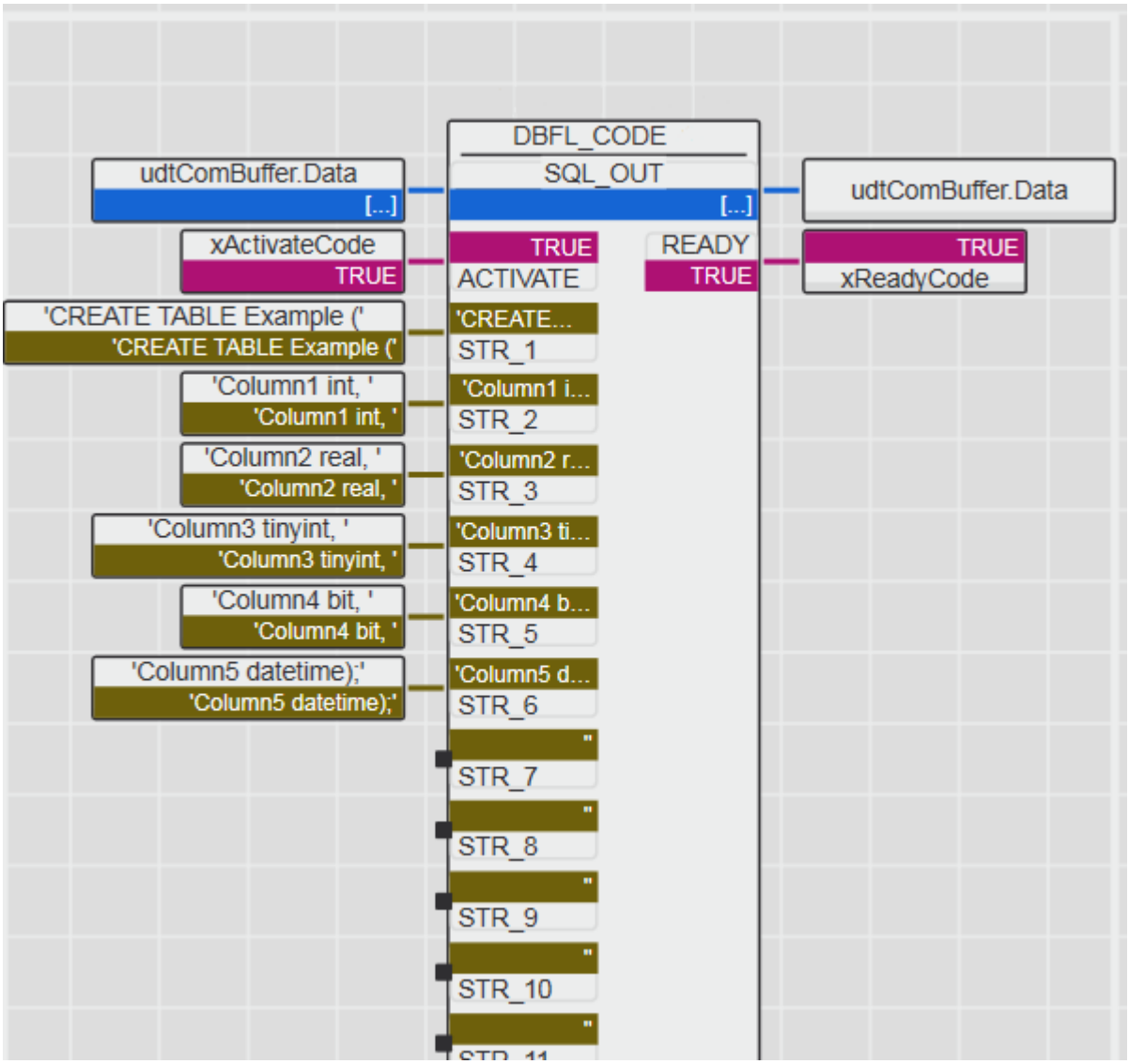
Initialize the variables of the DBFL_MySQL_ACCESS function block:



- Set the initialization value of the "strDB_User", "strDB_Password", "strIP_Adress", "strIP_Port" and "strDB_Name" variables with the data from your database.
- Activate the variable "xIP_Activate" of the DBFL_MySQL_ACCESS function block by double-clicking and overwrite. An attempt is made to establish a connection to the database.
- TCP_Ready and SQL_Ready are set to TRUE.

29.2.2 Creating a database table

Initialize the variables of the DBFL_CODE function block :



- Set the initialization value of the "STR_1" variable with the SQL command “CREATE TABLE Example (".
- Set the initialization value of the "STR_2" variable with the SQL command "Column1 int,".
- Set the initialization value of the "STR_3" variable with the SQL command "Column2 real,".
- Set the initialization value of the "STR_4" variable with the SQL command "Column3 tinyint,".
- Set the initialization value of the "STR_5" variable with the SQL command "Column4 bit,".
- Set the initialization value of the "STR_6" variable with the SQL command "Column5 datetime);".

Activate the variable "xActivateCode" of the DBFL_CODE function block by double-clicking and overwrite. For this action the TCP_Ready and SQL_Ready outputs of the DBFL_MySQL_ACCESS function block have to be TRUE.

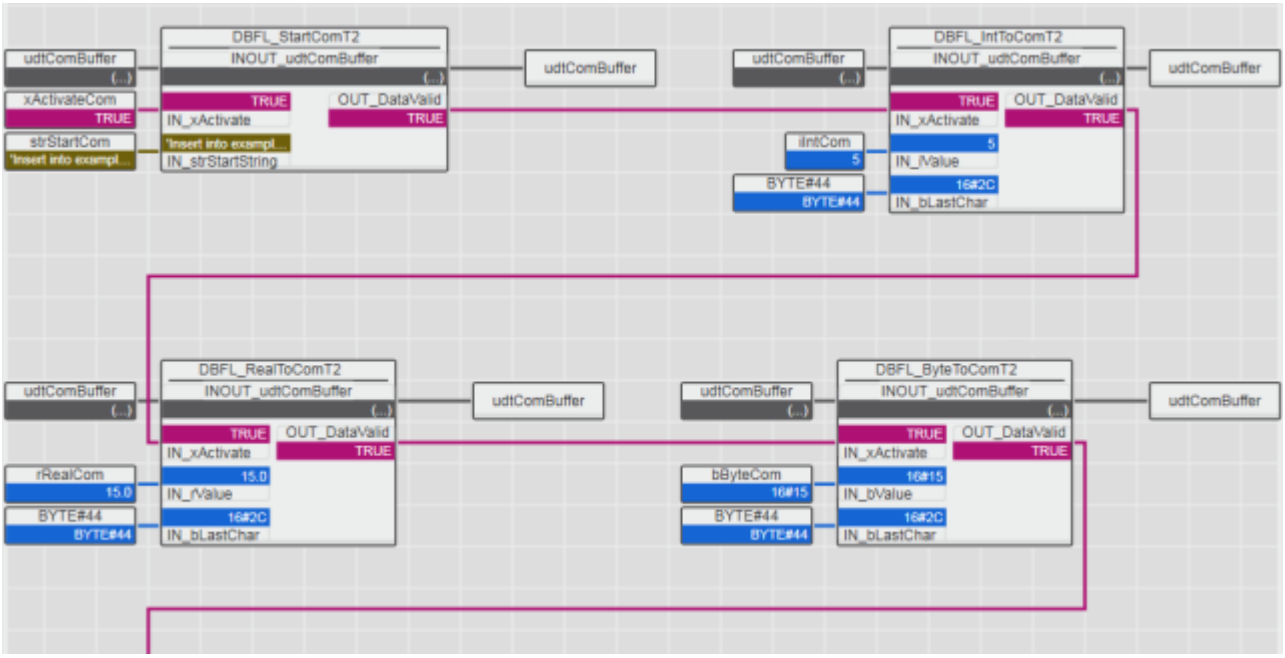
The SQL command is written to the udtComBuffer. Data is passed to the DBFL_MySQL_ACCESS function block. IF the xReadyCode output is TRUE, the xDB_Activate input of the DBFL_MySQL_ACCESS function block can be set to TRUE. If the transmission attempt is successful, the SQL_DONE output is set to TRUE at the DBFL_MySQL_Access function block.

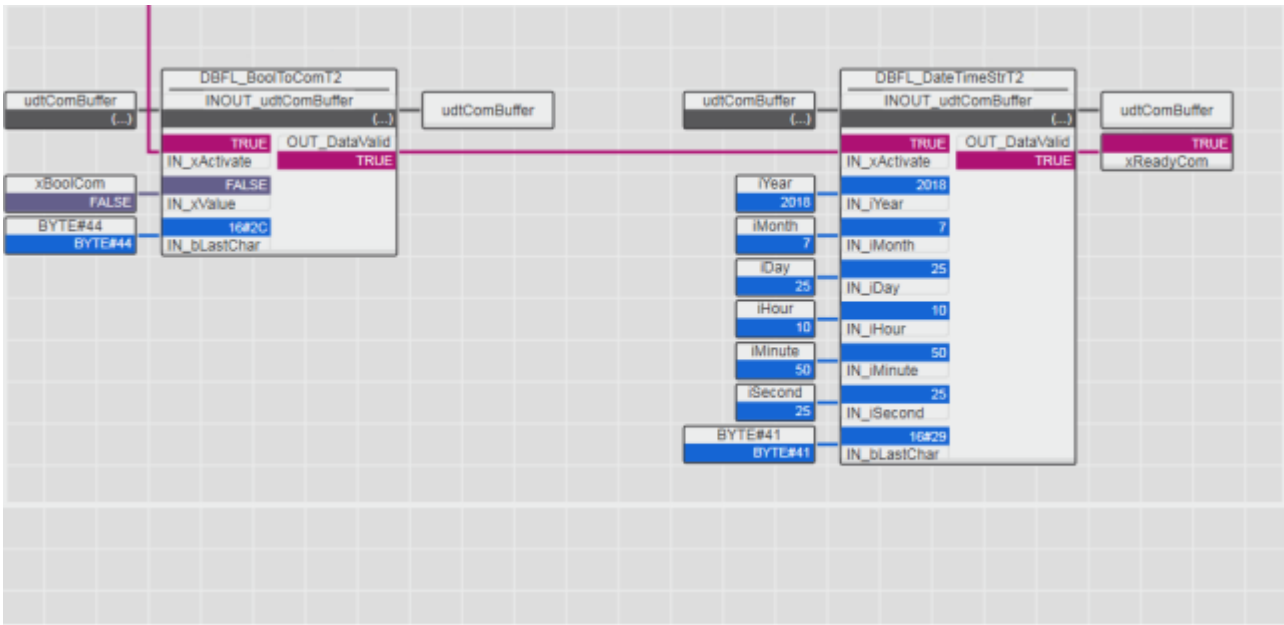
29.2.3 Writing to the database

Initialize the variable "strSQL_StartString" of the DBFL_StartComT2 function block with the SQL command "INSERT INTO Example VALUES (".

Initialize the value variables of the function blocks below:

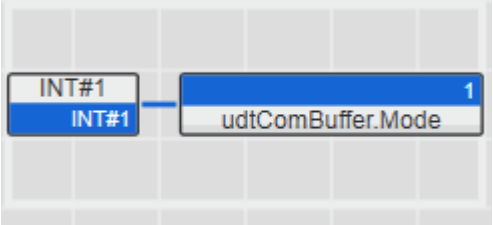
- DBFL_IntToComT2: The block inserts a value of type INT in the SQL command.
- DBFL_RealToComT2 The block inserts a value of type REAL in the SQL command.
- DBFL_ByteToComT2 The block inserts a value of type BYTE in the SQL command.
- DBFL_BoolToComT2 The block inserts a value of type BOOL in the SQL command.
- DBFL_DateTimeStrT2 The block inserts a date / time value in the SQL command.





Attention: To generate a MySQL compliant command, the udtComBuffer.Mode parameter must be assigned with the value INT#1

Example: MySqlCommand.Mode: = INT#1; (* MySqlCommand is of type DBFL_UDT_SQL_COMMAND *)



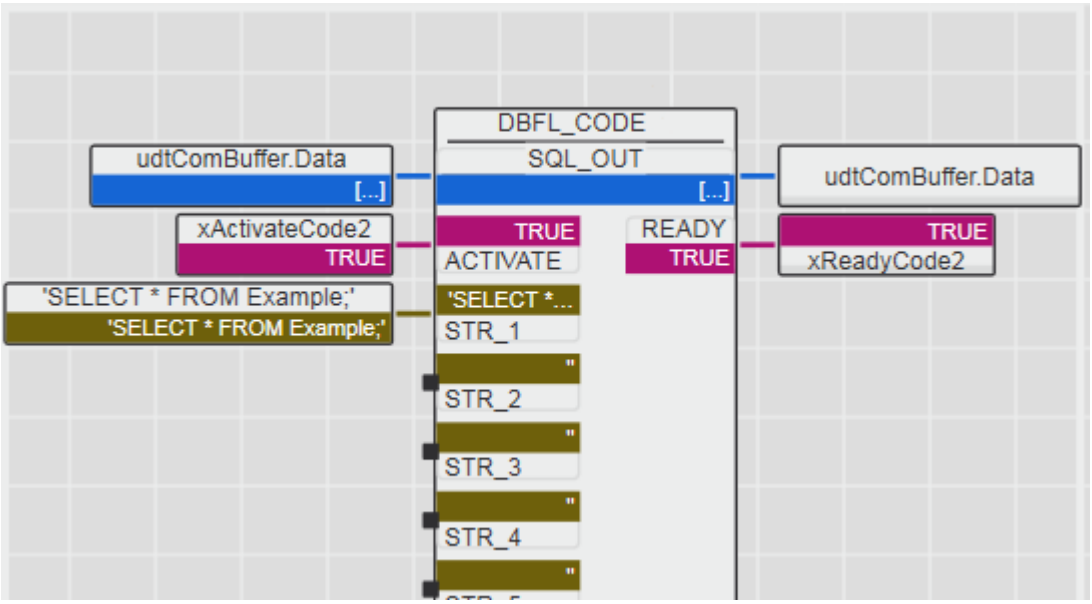
The IN_bLastChar input enables entering the separator and terminator between the individual values in the SQL command. BYTE#44 = , BYTE#41 =)

The SQL command is written to the udtComBuffer and passed to the DBFL_MySQL_ACCESS function block. Make sure that the variables are written into the command in the correct order. If the xReadyCode output of the last *ToCom block is TRUE, set the
xDB_Activate input of the DBFL_MySQL_ACCESS function block also to TRUE. If the transmission attempt is successful, the SQL_DONE output is set to TRUE at the DBFL_MySQL_Access function block.

29.2.4 Reading data from a table

To read data from database initialize the variable "STR_1" of the DBFL_CODE function block with the SQL command "SELECT * FROM Example". With this command you read all data of the database. If you only want to read certain columns, specify them in the command. For example: 'select column1, column3 from example'

Activate the variable "xActivateCode2" of the DBFL_CODE function block by double-clicking and overwrite.



The SQL command is wriiten to the udtComBuffer and passed to the DBFL_MySQL_ACCESS function block. If the transmission attempt is successful, the SQL_DONE output is set to TRUE at the DBFL_MySQL_Access function block.

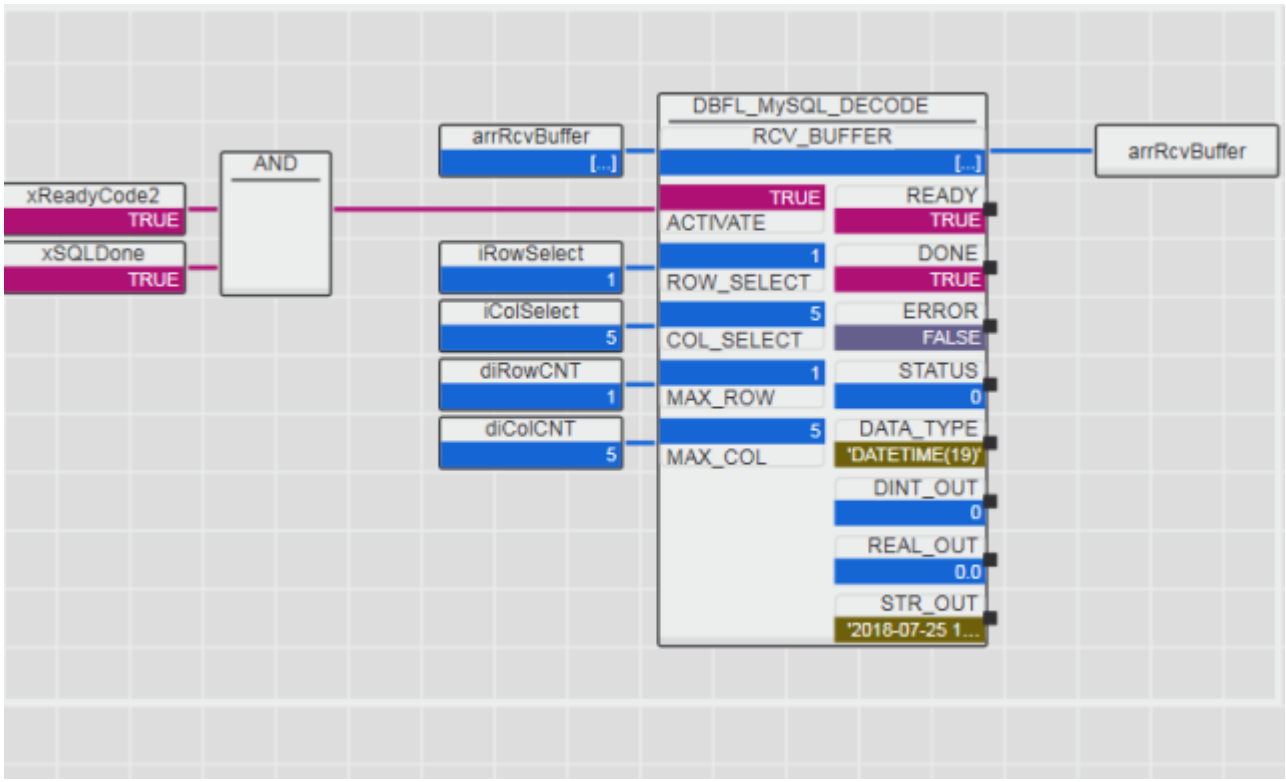
The data record returned by the database is decoded by the DBFL_MySQL_DECODE function block.

To select a row, set the "iRowSelect" parameter in the DBFL_MySQL_DECODE function block.

To select a column, set the "iColSelect" parameter in the DBFL_MySQL_DECODE function block.

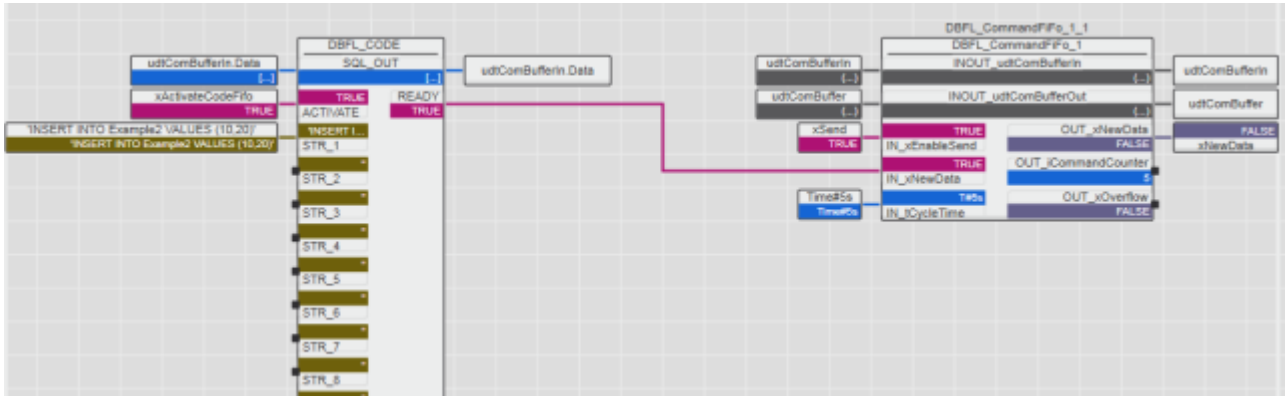
Activate the variable "xDecodeActivate" of the DBFL_MySQL_DECODE function block by double-clicking and overwrite.

The data for the returned cell is now available at the corresponding output. In the example, the data type of the cell is date time. The content is displayed at STR_OUT output.



29.2.5 CommandFiFo

An other option to send commands to the SQL database is the DBFL_CommandFiFo function block. With this function block you can save the commands and send them to the database (**F**irst **i**n **F**irst **o**ut)



For Example:

1. Activate the DBFL_Code function block
2. When the xReady of the DBFL_Code is True the xNewData of the CommandFiFo is set to TRUE
3. The command will be save in the buffer (iCommandCounter = 1)
4. Repeat step 1-3 a few times (iCommandCounter = x)
5. Activate the xEndableSend of the CommandFiFo
6. The commands are sent to the database according to the first in first out principle
7. With each new command, the output xNewData is set to True for one cycle
8. Connect output xNewData with the xDB_Activate input of the Access function block

30 Appendix

30.1 SQL data types

Data type	Designation	Size	Value range	Initialization value
BOOL	Boolean	1	TRUE/FALSE	0
SINT	8-bit integer	8	-128 to 127	0
INT	Integer	16	-32,768 to 32,767	0
DINT	Double integer	32	-2,147,483,648 to 2,147,483,647	0
USINT	8-bit unsigned integer	8	0 to 255	0
UINT	Unsigned integer	16	0 to 65,535	0
UDINT	Unsigned double integer	32	0 to 4,294,967,295	0
REAL	Floating-point number	32	-3.402823466 E+38 to -1.175494351 E-38 and +1.175494351 E-38 to +3.402823466 E+38	0.0
LREAL	Long floating-point number	64	-1.7976931348623158 E+308 to -2.2250738585072014 E-308 and +2.2250738585072014 E-308 to +1.7976931348623158 E+308	0.0
TIME	Time	32	0 to 4,294,967,295 ms	t#0 s
BYTE	8-bit string	8	0 to 255 (16#00 to 16#FF)	0
WORD	16-bit string	16	0 to 65,535 (16#00 to 16#FFFF)	0
DWORD	32-bit string	32	0 to 4,294,967,295 (16#00 to 16#FFFFFFFF)	0
STRING	84-bit string	84	Byte 0 - 1 offset to maximum length Byte 2 - 3 current length Byte 4 - 83 characters Byte 84 zero terminator	

30.2 Diagnosis of used firmware function blocks

30.2.1 TCP_SOCKET, TCP_RECEIVE, TCP_SEND

ERROR = FALSE

Status code	Description
16#0000	Situation is normal (no error).
16#8000	Socket is trying to connect the partner.
16#8001	Server is listening for a client.
16#8002	Server has rejected a client because the IP address and port number do not match.
16#8003	Not all data could be sent. Remaining data will be sent in the next cycle(s).
16#8004	Not all data received: Received length < Expected length

ERROR = TRUE

Error code	Description	Error only for
16#C001	Socket creation failed.	
16#C002	IP has wrong format.	
16#C003	Memory allocation failed.	
16#C100	Unexpected error during connecting of a client to a server.	TCP/TLS_SOCKET
16#C101	Unexpected error during receive operation.	UDP/TCP/TLS_RECEIVE
16#C102	Unexpected error during send operation.	UDP/TCP/TLS_SEND
16#C103	Unexpected error during bind operation.	UDP_SOCKET
16#C104	Unexpected error during listen operation.	TCP/TLS_SOCKET
16#C105	Unexpected error during accept operation.	TCP/TLS_SOCKET
16#C150	The TLS parameterization of the TLS_SEND/TLS_RECEIVE function blocks is inconsistent with the TLS_SOCKET function block. This is the case when: • TLS_SEND/TLS_RECEIVE require secure transmission/reception of data (SEND_SECURE/RECEIVE_SECURE input = TRUE), but the socket is not yet initialized for TLS communication (START_TLS input of TLS_SOCKET is FALSE). • TLS_SEND/TLS_RECEIVE require insecure transmission/reception of data (SEND_SECURE/RECEIVE_SECURE input = FALSE), but the socket is already initialized for TLS communication (START_TLS input of TLS_SOCKET is TRUE).	TLS_*
16#C151	An error regarding the START_TLS input of the TLS_SOCKET function block has occurred. START_TLS was set from TRUE to FALSE during opened TLS socket (ACTIVE input = TRUE). This is the case when:	TLS_*
16#C201	There are too many open sockets in the underlying socket provider.	
16#C202	An operation on a nonblocking socket cannot be completed immediately.	
16#C204	The datagram is too long.	

16#C205	Only one use of an address is normally permitted.	
16#C206	The selected IP address is not valid in this context.	
16#C207	The connection was aborted by the .NET Framework or the underlying socket provider.	
16#C208	The connection was reset by the remote peer.	
16#C210	The application tried to send or receive data, and the Socket is not connected (_SOCKET.ACTIVE == False).	
16#C211	No such host is known. The name is not an official host name or alias.	
16#C212	An unspecified System.Net.Sockets. Socket error has occurred.	
16#C213	The remote host is actively refusing a connection.	
16#C214	An invalid argument was supplied to a System.Net.Sockets.Socket member.	
16#C215	A blocking operation is in progress.	
16#C216	The overlapped operation was aborted due to the closure of the System.Net.Sockets.Socket.	
16#C217	The application has initiated an overlapped operation that cannot be completed immediately.	
16#C218	A blocking System.Net.Sockets.Socket call was canceled.	
16#C219	An attempt was made to access a System.Net.Sockets.Socket in a way that is forbidden by its access permissions.	
16#C21A	An invalid pointer address was detected by the underlying socket provider.	
16#C21B	A System.Net.Sockets.Socket operation was attempted on a non-socket.	
16#C21C	A required address was omitted from an operation on a System.Net.Sockets.Socket.	
16#C21D	An unknown, invalid, or unsupported option or level was used with a System.Net.Sockets.Socket.	
16#C21E	The protocol type is incorrect for this System.Net.Sockets.Socket.	
16#C21F	The protocol is not implemented or has not been configured.	
16#C220	The support for the specified socket type does not exist in this address family.	
16#C221	The address family is not supported by the protocol family.	
16#C222	The protocol family is not implemented or has not been configured.	
16#C223	he address family specified is not supported. This error is returned if the IPv6 address family was specified and the IPv6 stack is not installed on the local machine. This error is returned if the IPv4 address family was specified and the IPv4 stack is not installed on the local machine.	
16#C224	The network is not available.	
16#C225	No route to the remote host exists.	
16#C226	The application tried to set System.Net.Sockets.SocketOptionName. KeepAlive on a connection that has already timed out.	
16#C227	No free buffer space is available for a System.Net.Sockets.Socket operation.	
16#C228	A request to send or receive data was disallowed because the System.Net.Sockets.Socket has already been closed.	
16#C229	The connection attempt timed out, or the connected host has failed to respond.	
16#C22A	The operation failed because the remote host is down.	

16#C22B	There is no network route to the specified host. Could not connect to DEST_IP.	
16#C22C	Too many processes are using the underlying socket provider.	
16#C22D	The network subsystem is unavailable.	
16#C22E	The version of the underlying socket provider is out of range.	
16#C22F	The underlying socket provider has not been initialized.	
16#C230	A graceful shutdown is in progress.	
16#C231	The specified class was not found.	
16#C232	The name of the host could not be resolved. Try again later.	
16#C233	The error is unrecoverable or the requested database cannot be located.	
16#C234	The requested name or IP address was not found on the name server.	

30.3 Data types

```
TYPE
(*Array of Bytes*)
    DBFL_ARR_BYTE_0_9     : ARRAY[0..0009]  OF BYTE;
    DBFL_ARR_UNIQUE       : ARRAY[0..0015]  OF BYTE;
    DBFL_ARR_BYTE_0_19    : ARRAY[0..0019]  OF BYTE;
    DBFL_ARR_BYTE_0_80    : ARRAY[0..0080]  OF BYTE;
    DBFL_ARR_BYTE_0_1036  : ARRAY[0..1036]  OF BYTE;
    (*MySQL*)
    DBFL_ARR_BYTE_0_1439  : ARRAY[0..1439]  OF BYTE;
    (*MsSQL*)
    DBFL_ARR_BYTE_0_1459  : ARRAY[0..1459]  OF BYTE;
    (* MSSQL receive buffer [7 * 1460] *)
    DBFL_ARR_BYTE_0_10219 : ARRAY[0..10219] OF BYTE;
    DBFL_ARR_GRAPH        : ARRAY[0..1461]  OF BYTE;
(*Internal*)
    DBFL_ARR_DWORD_0_79   : ARRAY[0..0079] OF DWORD;

    DBFL_UDT_STRING255    : STRING(255);
    DBFL_UDT_STRING20     : STRING(20);

    (*Array for Deode fb*)
    DBFL_COLUMN : STRUCT
        Typ     : USINT;
        x       : BYTE;
        y       : BYTE;
    END_STRUCT;
    DBFL_ARR_COLUMN : ARRAY[0..255] OF DBFL_COLUMN;

    (*SQL_IN*)
    DBFL_UDT_SQL_COMMAND : STRUCT
        Data : DBFL_ARR_BYTE_0_1439; (*SQL Query*)
        WP   : INT;
        Mode : INT; (* 1 = MySQL; all others MSSQL *)
    END_STRUCT;

    (*ARRAY for SQL-Querys*)
    DBFL_ARR_SQLCOM_FIFO    : ARRAY[0..49] OF DBFL_ARR_BYTE_0_1439;

    DBFL_UDT_CopyBuff : STRUCT (*Buffer for TSQL_Decode*)
        Byte0   : BYTE;
        Byte1   : BYTE;
        Byte2   : BYTE;
        Byte3   : BYTE;
        Byte4   : BYTE;
        Byte5   : BYTE;
        Byte6   : BYTE;
        Byte7   : BYTE;
        RealWert : REAL; (*REAL_OUT*)
    END_STRUCT ;

    (*Server Info for MySQL*)
    DBFL_UDT_MySQL_Server_Info : STRUCT
        OldPW_Handling   : BOOL;
        DemoModeActive   : BOOL;(*deactivated for PCWE*)
        TimeToEndDemo    : TIME;
        Protocol_Version : USINT;
        Server_Language  : USINT;
```

```
        Server_Version   : STRING;
        Thread_Number    : UDINT;
        Server_Caps      : WORD;
        Server_Status    : WORD;
        Scramble_Buff    : DBFL_ARR_BYTE_0_19;
    END_STRUCT;
    (*Server Info for MsSQL*)
    DBFL_UDT_MSSQL_Server_Info : STRUCT
        DemoModeActive : BOOL; (*deactivated for PCWE*)
        TimeToEndDemo  : TIME;
        ServerName     : STRING;
        ServerVersion  : STRING;
    END_STRUCT;

END_TYPE
```

31 Support

For technical support please contact your local PHOENIX CONTACT agency

at https://www.phoenixcontact.com

Owner:

PHOENIX CONTACT Electronics GmbH
Business Unit Automation Systems
System Services
Library Services