SQL

straton user guide – Rev. 6

sales@straton-plc.com

CP_R31_006_H

straton



STRATON AUTOMATION, All Rights Reserved

The information contained in this document is the property of STRATON AUTOMATION. The distribution and/or reproduction of all or part of this document in any form whatsoever is authorized only with the written authorization of STRATON AUTOMATION. The technical data are used only for the description of the product and do not constitute a guarantee of quality in the legal sense of the term. We reserve the right to make technical changes.

Content

1. PREREQUISITE 4

2. MYSQL 4
   2.1. MySQLConnect 4
      2.1.1. Inputs 5
      2.1.2. Outputs 5
   2.2. MySQL Secure 5
      2.2.1. Overview 5
      2.2.2. Inputs 6
      2.2.3. Output 6
   2.3. MYSQLQuery / MYSQLQueryTXB 6
      2.3.1. Overview 6
      2.3.2. Inputs 7
      2.3.3. Outputs 7
      2.3.4. Result 8

3. SQLITE 8
   3.1. SqlLite Opening a database 8
      3.1.1. Inputs 9
      3.1.2. Outputs 9
   3.2. SqLite / SqLiteQuery Txb 9
      3.2.1. Inputs 10
      3.2.2. Outputs 10
      3.2.3. Result 11

4. ERRORS 11

5. SAMPLE APPLICATIONS 12
   5.1. Configure a secure connection 12
   5.2. Push a variable in the SQL database 13
   5.3. Create a table 14
   5.4. Commit / Rollback 15

1. Prerequisite

Warning: SQLite and MySQL function blocks are optional features and may be not supported by your runtime. If you try to download a project using such blocks on a runtime which does not support them, you will get some errors in the output window specifying that some embedded software is missing. Refer to your OEM for further information about the runtime.

2. MySQL

All interactions with MySQL database are done through function blocks, there is no fieldbus configurator for it.

On a Windows runtime, the following DLLs must be present in the runtime's folder: *T5BlocksMYSQL.dll* and *libmysql.dll*. The first one is provided by COPA-DATA. The second one can be found in the "lib" folder of the mysql-connector (win32, not win 64! Because T5 run in 32 bit) ZIP Archive, available here:

https://downloads.mysql.com/archives/c-c/

(Tested by CDFR with the version 6.0.2 and 6.1.11)

There are 4 blocks for Mysql:

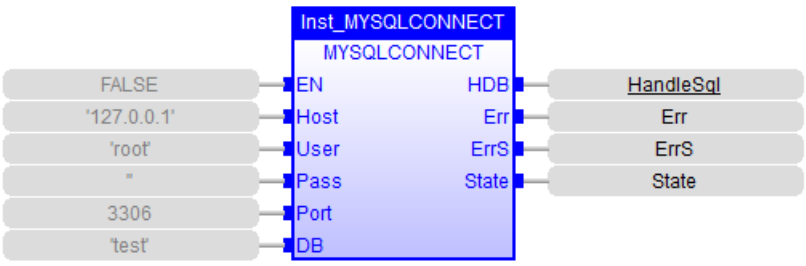
- MYSQLCONNECT
- MYSQLSECURE
- MYSQLQUERY
- MYSQLQUERYTXB

2.1. MySQLConnect

This block opens a connection to a remote or local MySql server. It also select the database that will be used.

Function Block: MYSQLCONNECT

Example of use in FBD Language:



2.1.1. Inputs

EN	*BOOL*	Connect to or disconnect from a database. The connection is established on a rising edge of the EN input. It is disconnected on a falling edge of EN.
HOST	*STRING*	URL or IP address of a MYSQL server
USER	*STRING*	User name for login
PASS	*STRING*	Password name for login
PORT	*DINT*	Ethernet port number
DB	*STRING*	Name of the database to use

2.1.2. Outputs

HDB	*DINT*	Handle of the connection or 0 if not connected
ERR	*DINT*	Code of the last SQL error*
ERRS	*STRING*	Description of the last error*
STATE	*STRING*	Description of the last MySQL state report code

* The error code and description are sent from the server. See chapter 4.

All string variables are limited to 255 characters. Any character may be entered. Use the '$' prefix for special characters, such as explained in the straton Help.

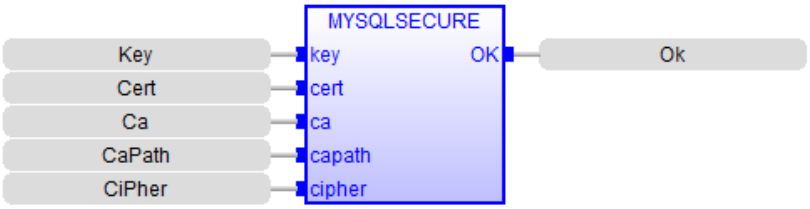
2.2. MySQL Secure

2.2.1. Overview

It is not mandatory to use it. If not used, the connection will not be secured.

If used, **it must be called before MysqlConnect** block.

Using it allows the connection to be secured through the SSL encryption supported by Mysql.



2.2.2. Inputs

KEY	*STRING*	Pathname of the key file
CERT	*STRING*	Pathname of the certificate file
CA	*STRING*	Pathname of the certificate authority file
CAPATH	*STRING*	Pathname to a directory that contains trusted SSL CA certificates in PEM format
CIPHER	*STRING*	List of permissible ciphers to use for SSL encryption

2.2.3. Output

OK	BOOL	TRUE if successful

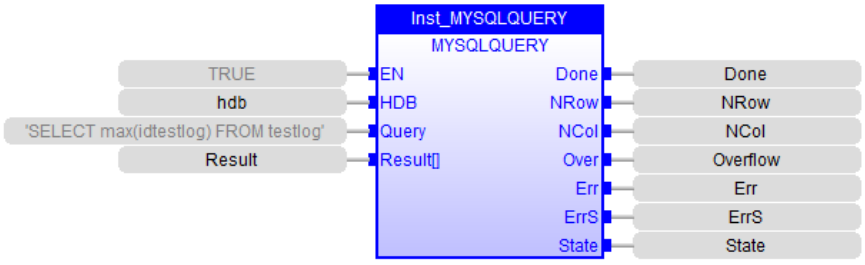
2.3. MYSQLQuery / MYSQLQueryTXB

2.3.1. Overview

These blocks send a unique query (multiple simultaneous query are not supported) to the currently active database on the server that corresponds to the connection handle (HCnx).

It then readout the results, and return an array filled with values.

The query can be anything supported by the database (for example SELECT, INSERT, UPDATE, CREATE TABLE, START TRANSACTION, COMMIT, ROLLBACK, USE, SHOW, whatever supported by the database.). There is no limitation in the type of query, every query will be forwarded "as is" to the database engine.



	MySql
Short Query (<=255 characters)	MysqlQuery
Long Query (>255 characters)	MysqlQueryTXB

Strings are limited to 255 characters in straton, so for longer queries, it is needed to use MySqlQueryTxb that use a textbuffer instead of a string input.

2.3.2. Inputs

EN	*BOOL*	Trigger a query (see notes)
HCnx / HDB	*DINT*	Handle of a database (from respectively mysqlConnect or sqliteDatabase block)
QUERY *	*STRING*	SQL query passed as a string
QUERYTXB **	*DINT*	Handle of a text buffer containing the SQL query
RESULT	*STRING[ ]*	Array of strings filled on output with the result of the query Note: it is connected as an input, but behave as an output (as long as it is an array – Check IN_OUT in the documentation for more details on this). See the *Result* part of this chapter.

** For the MySQLQuery the query is a String (limited to 255 characters)*

*** For the MySQLQueryTXB blocks, the query is a handle of a "text buffer", an object that allow text longer than 255 characters. The option "Allow dynamic memory allocation" of the project should be set to allow its use. Refer to online help for details about text buffer use.*

2.3.3. Outputs

DONE	*BOOL*	TRUE during 1 cycle when a query finish (whatever the Error is 0 or not). FALSE the rest of the time. Stays FALSE if the HCnx or HDB is incorrect
NROW	*DINT*	Full number of rows returned by the query
NCOL	*DINT*	Full number of columns returned by the query
OVER	*BOOL*	TRUE if the RESULT array is too small, so last returned cells were lost
ERR	*DINT*	Code of the last SQL error (by MySQL server)
ERRS	*STRING*	Description of the last error (by MySQL server)
STATE	*STRING*	Description of the last MySQL state report code (by MySQL server)

** Only one query can be run at the same time. Next queries are queued and will be executed afterward. See error code -2 in chapter 4.*

2.3.4. Result

The array storing the result can be 1-dimension or 2-dimensions. The choice is made while declaring the array in the list of variables.

The blocks automatically adapt to the dimension of the array.

1-dimension : example with **Result[15]**	
Result3	
Result3[0]	'Basseterre'
Result3[1]	'Saint Kitts and Nevis'
Result3[2]	'Constitutional Monarchy'
Result3[3]	'Creole English'
Result3[4]	'100.0'
Result3[5]	'Port-au-Prince'
Result3[6]	'Haiti'
Result3[7]	'Republic'
Result3[8]	'Haiti Creole'
Result3[9]	'100.0'
Result3[10]	'Carrefour'
Result3[11]	'Haiti'
Result3[12]	'Republic'
Result3[13]	'Haiti Creole'
Result3[14]	'100.0'

2-dimensions : example with **Result[3 , 5]**	
Result3	
Result3[0]	
Result3[0,0]	'Basseterre'
Result3[0,1]	'Saint Kitts and Nevis'
Result3[0,2]	'Constitutional Monarchy'
Result3[0,3]	'Creole English'
Result3[0,4]	'100.0'
Result3[1]	
Result3[1,0]	'Port-au-Prince'
Result3[1,1]	'Haiti'
Result3[1,2]	'Republic'
Result3[1,3]	'Haiti Creole'
Result3[1,4]	'100.0'
Result3[2]	
Result3[2,0]	'Carrefour'
Result3[2,1]	'Haiti'
Result3[2,2]	'Republic'
Result3[2,3]	'Haiti Creole'
Result3[2,4]	'100.0'

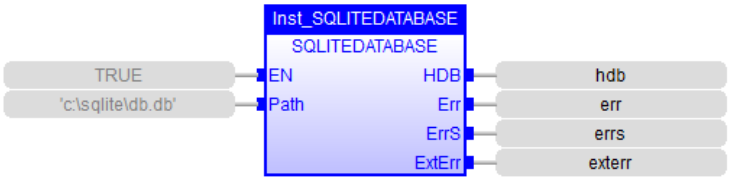
3. SQLite

On a Windows runtime, the T5Blockssqlite.dll, provided by COPA-DATA, must be present in the runtime's folder.

3.1. SqlLite Opening a database

This block opens a SQLite database. The database is opened on a rising edge of the EN input. It is created if not existing. It is closed on a falling edge of EN.

Function Block: SQLITEDATABASE



3.1.1. Inputs

EN	*BOOL*	Connect to or disconnect from a database. The connection is established on a rising edge of the EN input. It is disconnected on a falling edge of EN.
PATH	*STRING*	Pathname of the database file

3.1.2. Outputs

HDB	*DINT*	Handle of the connection or 0 if not connected
ERR	*DINT*	Code of the last SQL error*
ERRS	*STRING*	Description of the last error*
EXTERR	*DINT*	Extended SQLite error code

** The error code and description are sent from the server. See chapter 4.*

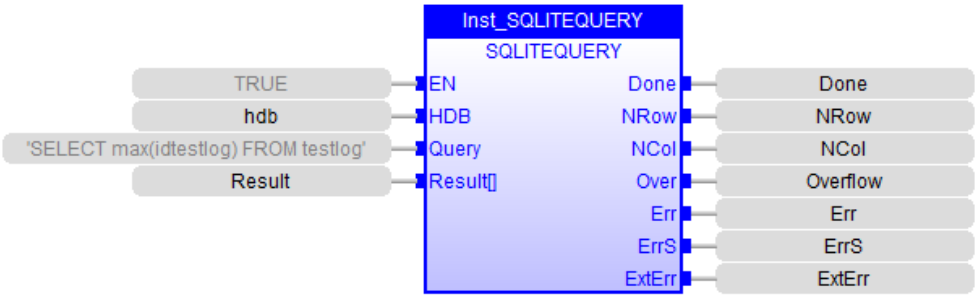
All strings are limited to 255 characters. If the file specified by PATH does not exist, then the database is open for creation.

3.2. SqLite / SqLiteQuery Txb

These blocks sends a unique query (multiple simultaneous query are not supported) to the currently active database on the database that corresponds to the connection handle (HCnx).

It then readout the results, and return an array filled with values.

The query can be anything supported by the database (for example SELECT, INSERT, UPDATE, CREATE TABLE, START TRANSACTION, COMMIT, ROLLBACK, USE, SHOW, whatever supported by the database.). There is no limitation in the type of query, every query will be forwarded "as is" to the database engine.



	SQLite
Short Query (<=255 characters)	SQLiteQuery
Long Query (>255 characters)	SQLiteQueryTXB

3.2.1. Inputs

EN	*BOOL*	Trigger a query (see notes)
HCnx / HDB	*DINT*	Handle of a database (from respectively mysqlConnect or sqliteDatabase block)
QUERY *	*STRING*	SQL query passed as a string
QUERYTXB **	*DINT*	Handle of a text buffer containing the SQL query
RESULT	*STRING[ ]*	Array of strings filled on output with the result of the query Note: it is connected as an input, but behave as an output (as long as it is an array – Check IN_OUT in the documentation for more details on this). See the *Result* part of this chapter.

** For the SQLiteQuery blocks, the query is a String (limited to 255 characters)*

*** For the SQLiteQueryTXB blocks, the query is a handle of a "text buffer", an object that allow text longer than 255 characters. The option "Allow dynamic memory allocation" of the project should be set to allow its use. Refer to online help for details about text buffer use.*

3.2.2. Outputs

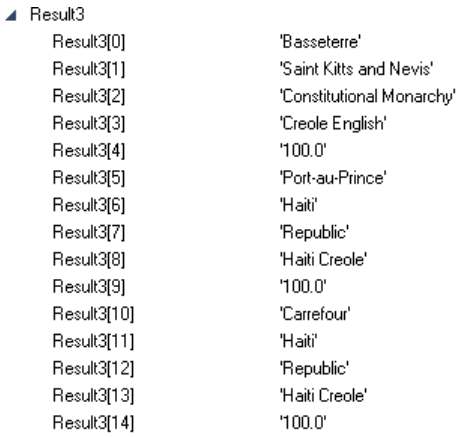
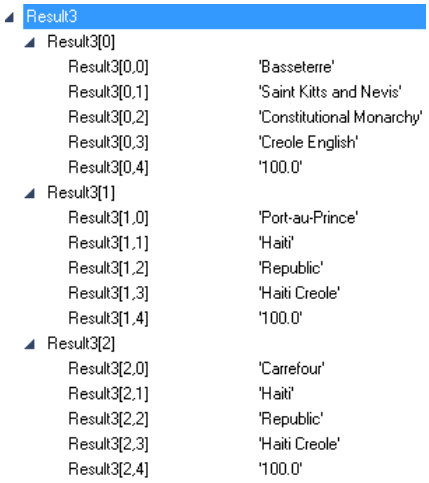
DONE	*BOOL*	TRUE during 1 cycle when a query finish (whatever the Error is 0 or not). FALSE the rest of the time. Stays FALSE if the HCnx or HDB is incorrect
NROW	*DINT*	Full number of rows returned by the query
NCOL	*DINT*	Full number of columns returned by the query
OVER	*BOOL*	TRUE if the RESULT array is too small, so last returned cells were lost
ERR	*DINT*	Code of the last SQL error (from SQLite)
ERRS	*STRING*	Description of the last error (from SQLite)
ERREXT	*DINT*	Extended error code of the last SQL error (from SQLite)

** Only one query can be run at the same time. Next queries are queued and will be executed afterward. See error code -2 in chapter 4.*

3.2.3. Result

The array storing the result can be 1-dimension or 2-dimensions. The choice is made when declaring the array in the list of variables.

The blocks automatically adapt to the dimension of the array.

1-dimension : example with **Result3[15]**	2-dimensions : example with **Result3[3 , 5]**
	

4. Errors

The errors are displayed in two formats: a DINT (double integer) and a STRING containing a short description of the error.

Possible errors are:

Greater than 0	The error has been generated by the SQL server. Check the server documentation to see what is the error : MySQL : https://dev.mysql.com/doc/refman/5.5/en/error-messages-server.html SQLite : https://www.sqlite.org/rescode.html
0	No Error
-1	The specified handle parameter is incorrect
-2	Another query is running. This query is pending and will be executed when the server will be ready.

Only errors -1 and -2 are triggered by straton. All others come from the SQL server.

5. Sample applications

This chapter describes some sample programs to be re-used as template to create an application.

Please note that the SQL FBs are not blocking and that a program should not block the PLC cycle, so the programmer needs to consider this behavior when designing a program.

Note: For one connection only one query at a time is possible.

The following samples are done for one specific SQL database, some adaptations can be required when used with another SQL database as the Query syntax could be different.

5.1. Configure a secure connection

The code configures the SQL security options and then connects to the mySql database

Major ST variables:

```
Handle : DINT; (* identifies the SQL connection*)
```

ST Code:

```
IF Handle = 0 THEN   (* If the handle is invalid, we try to connect *)
    MYSQLSECURE( '/home/straton/client-key.pem',  //private keys
                 '/home/straton/client-cert.pem', //certificate
                 '/home/straton/ca.pem',     //Certificate authority
                 '/home/straton/ListOfTrustedCertificates/',
                 'DHE-RSA-AES128-GCM-SHA256' );
    Inst_MYSQLCONNECT ( TRUE, '127.0.0.1', 'root', '', 3306, 'test' );
    Handle := Inst_MYSQLCONNECT.HDB;
    (* if a connection error occurs then the error output of the CONNECT
       block will be raised *)
END_IF;
```

If you need more details about MySQL secure connections check:

https://dev.mysql.com/doc/refman/5.7/en/creating-ssl-rsa-files.html

5.2. Push a variable in the SQL database

This application opens a SQL connection and sends query on every change of the value in order to log the value.

Major ST variables:

```
Handle : DINT; (* identifies the SQL connection*)
Inst_MYSQLCONNECT : MYSQLCONNECT;
Inst_MYSQLQUERY : MYSQLQUERY;
Inst_PLS : PLS; (* pulse train *)
```

ST code:

```
// Sample code to log a data value
//Connection to a database
IF Handle = 0 THEN    (* If the handle is invalid, we try to connect *)
    Inst_MYSQLCONNECT ( TRUE, '127.0.0.1', 'root', '', 3306, 'test' );
    Handle := Inst_MYSQLCONNECT.HDB;
    (* if a connection error occurs then the error output of the CONNECT
       block will be raised *)
END_IF;
(* Example of periodic insert*)
(* a variable X changes every 3 seconds *)
Inst_PLS ( TRUE, T#3s );
IF Inst_PLS.Q THEN
   x := ( x + 0.1 );
END_IF;
(* We convert to string to be usable in a SQL query *)
sX := ANY_TO_STRING(x);
sSinX := ANY_TO_STRING(sin(x));
// ...
  Inst_MYSQLQUERY (
NOT Inst_PLS.Q, (* will generate a rising edge on every 3 seconds *)
      Handle,
     'INSERT INTO test.testlog VALUES($'$',$''+ sX+' $',$'' +sSinX+'$');',
(*The $ is for escaping the ' that is a string delimiter*)
      ResultPerioInsert );
    LastInsertedX := x;
// End program
```

5.3. Create a table

This application opens a SQL connection, and creates a table.

Major ST variables:

```
Handle : DINT; (* identifies the SQL connection*)
Inst_MYSQLCONNECT : MYSQLCONNECT;
Inst_MYSQLQUERYTXB : MYSQLQUERYTXB;
queryTXB : DINT; (* handle of a valid text buffer *)
bTableCreated : BOOL; (* initially FALSE – set to TRUE when table created *)
```

ST code:

```
// Sample code to create a table
//Connection to a database
IF Handle = 0 THEN
(* If the handle is invalid, we try to connect *)
Inst_MYSQLCONNECT ( TRUE, '127.0.0.1', 'root', '', 3306, 'test' );
          Handle := Inst_MYSQLCONNECT.HDB;
END_IF;
//We initiate the TXB manager to have Text Buffer working for long queries
//Refer to the straton help for a description of any TXB… function or block
Inst_TxbManager (  );
//The query is longer than 255 characters, so we append string to a text buffer string by
string
queryTXB := TxbNewString ( 'CREATE TABLE `table01` (' );
TxbAppend (queryTXB, '`idtable` bigint(11) NOT NULL AUTO_INCREMENT,' );
TxbAppend (queryTXB, '`timestamp` int(11) DEFAULT NULL,');
TxbAppend (queryTXB, '`offset` int(11) DEFAULT NULL,');
TxbAppend (queryTXB, '`value` float DEFAULT NULL,');
TxbAppend (queryTXB, 'PRIMARY KEY (`idtable`)');
TxbAppend (queryTXB, ') ENGINE=InnoDB DEFAULT CHARSET=latin1;');

//We send the query if the table is not created yet
Inst_MYSQLQUERYTXB ((NOT bTableCreated & ( Handle <> 0 ) ), Handle, queryTXB, ResultOsTable
);
// ...
//We only want to create the table once, so we set bTableCreated to true
IF Inst_MYSQLQUERY.Done THEN
    bTableCreated := TRUE; // now the table is created
END_IF;
TxbFree (queryTXB); // release the text buffer
// End program
```

5.4. Commit / Rollback

This application opens a SQL connection, inserts some data using three query, then depending on the Boolean value '`ShallWeCommit`' the changes are commit or rollback.

ST code for ROLLBACK:

```
// Sample code for transaction with rollback
Inst_MYSQLQUERY_Start ( //start of a transaction
GOTransaction,
Handle,
'START TRANSACTION;',
ResultStartT );
Inst_MYSQLQUERY_Insert ( //insertion of data
Inst_MYSQLQUERY_Start.Done, //send query when previous query is done
Handle,
'INSERT INTO test.table01 VALUES($'10012$',$'1$',$'0.3$');',
ResultInsT );
Inst_MYSQLQUERY_Select ( //test that the data has been inserted
Inst_MYSQLQUERY_Insert.Done,
Handle,
'SELECT * FROM test.table01;',
ResultSelT );
NRow1 := Inst_MYSQLQUERY_Select.NRow; //number of rows returned
//NRow1 will be equal to 1, because we just inserted a row
IF ShallWeCommit = TRUE THEN
   WhatToDo := 'COMMIT;';
ELSE
   WhatToDo := 'ROLLBACK;';
END_IF;
Inst_MYSQLQUERY_Rollback_Commit ( //rollback or commit
Inst_MYSQLQUERY_Select.Done,
Handle,
WhatToDo,
ResultWtd );
Inst_MYSQLQUERY_Check ( //test what happened
Inst_MYSQLQUERY_Rollback_Commit.Done,
Handle,
'SELECT * FROM test.table01;',
ResultSelR );
NRow2 := Inst_MYSQLQUERY_Check.NRow; //number of rows returned
//NRow2 will be equal to 0 if the insert was canceled
//NRow2 will be equal to 1 if the insert has been committed
//End program
```