1. **Installation**

To install the component, please run Setup.exe. The installation directory is predetermined by the setup, C:\Programme\Kostal\Linecontroller DLL 2004. The documentation and the test program are automatically created entries in the Windows start menu. By this you can access this documentation and read more instructions during installation.

Following files will be installed:

|  |  |  |
| --- | --- | --- |
| Linecontroller.dll | Version 1.4.2 | DLL file |
| wedding.sql | — | SQL backup of the Wedding Table\* |
| article.sql | — | SQL backup of the article table \* |
| Linecontrolling.ini | Version 1.0 | Configuration file for the DLL chaining |
| Dokumentation.doc | Version 1.4.2 | The documentation |
| unins000.exe | Version 51.52 | Uninstall program |
| unins000.dat | — | Uninstall information |
| msado28.tlb | — | Uninstall information |

\* These files are not allowed be changed in any case, since after a change, no correct function can be guaranteed

1. **Library**
   1. **Duties of the library**

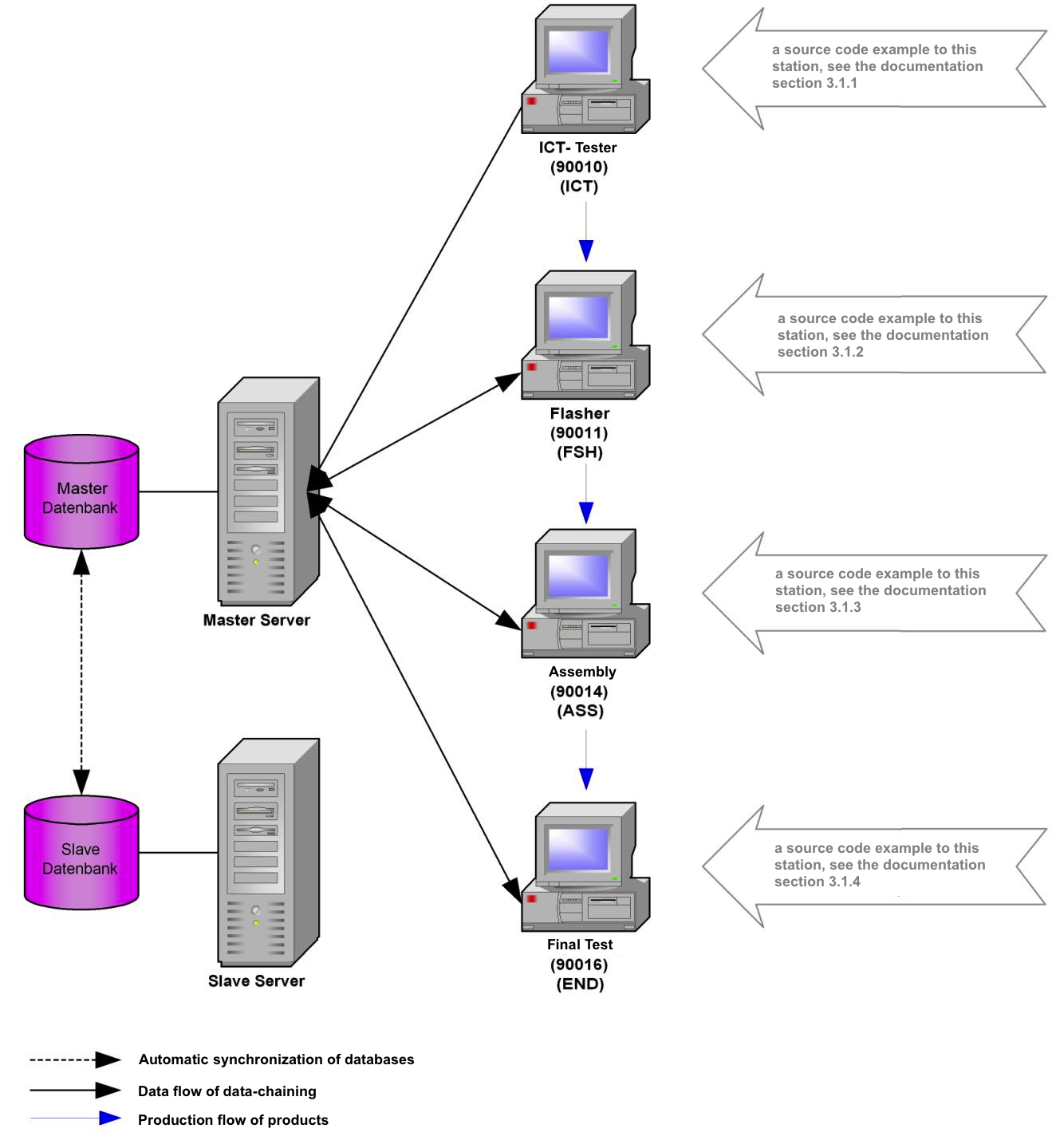
The Line Controller 2004 ActiveX DLL provides the possibility to write data into a SQL database chaining and read them from there. The special feature of this new version of the Line Controller DLL is that the user has two possible modes of available operation to their guests. Firstly, it is possible to store the data in a database, where can read again. However, this solution leads to the fact that when a database failures, not linked products can be produced. For such a case, the appropriate testing chaining must be issued. The second solution is based on a master-slave database, which virtually guarantees 99.9% availability for the data, since in case of a loss of the master database, the slave database takes over the tasks of the master database automatically. This solution is the clearly favored solution and should to possibility be used. The DLL can be integrated as a flexible component in all projects.

1. **Examples**

For the Line Controller 2004 ActiveX DLL, there are two application examples. These two examples will be discussed in more detail in the following two pages. The graph respectively shows only connections to the master database server, because the second server for the DLL is transparent and is only used when the first crash.

* 1. **Chaining of a product**

The following diagram shows the linkage of the Line Controller 2004 ActiveX DLL from a product that goes through several stages of production. For each production step, it will check whether it has been good tested on the previous station. As described in the diagram, you can see a source code example for each station shown in the documentation mentioned points. It should be noted that they are examples and not requirements. Everyone is responsible for the correct program functions themselves.



* + 1. **Example source code for ICT tester**

LineControl.SerialNo = “0010302140230234“

LineControl.ArticleNo = “00103021“

LineControl.TestResult = True

LineControl.CurrentTest = “ICT“

LineControl.TraceID = 90010

LineControl.AdditionalText(0) = “Resistor R27 = 56 OHM“

If LineControl.WriteCurrentStamp() = 0 Then

MsgBox “Record written “

Else

MsgBox “Record not written “

End If

* + 1. **Example source code for flasher**

LineControl.SerialNo = "0010302140230234"

LineControl.ArticleNo = "00103021"

LineControl.PreviousTest (0) = "ICT"

LineControl.SearchMode = 1

If LineControl.ReadPreviousStamp () = 0 Then

MsgBox "Good tested part"

else

MsgBox "Poor tested part"

Exit Sub

End If

LineControl.SerialNo = "0010302140230234"

LineControl.ArticleNo = "00103021"

LineControl.TestResult = TRUE

LineControl.CurrentTest = "FSH"

LineControl.TraceID = 90011

If LineControl.WriteCurrentStamp () = 0 Then

MsgBox "Record written"

else

MsgBox "Record not written"

End If

* + 1. **Example source code for assembly**

LineControl.SerialNo = "0010302140230234"

LineControl.ArticleNo = "00103021"

LineControl.PreviousTest (0) = "FSH"

LineControl.SearchMode = 1

If LineControl.ReadPreviousStamp () = 0 Then

MsgBox "Good tested part"

else

MsgBox " Poor tested part "

Exit Sub

End If

LineControl.SerialNo = "0010302140230234"

LineControl.ArticleNo = "00103021"

LineControl.TestResult = TRUE

LineControl.CurrentTest = "ASS"

LineControl.TraceID = 90014

If LineControl.WriteCurrentStamp () = 0 Then

MsgBox " Record written "

else

MsgBox "Record not written"

End If

* + 1. **Example source code for final test**

LineControl.SerialNo = "0010302140230234"

LineControl.ArticleNo = "00103021"

LineControl.PreviousTest (0) = "ICT"

LineControl.PreviousTest (1) = "FSH"

LineControl.PreviousTest (2) = "ASS"

LineControl.SearchMode = 3

If LineControl.ReadPreviousStamp () = 0 Then

MsgBox " Good tested part "

else

MsgBox " Poor tested part "

Exit Sub

end if

LineControl.SerialNo = "0010302140230234"

LineControl.ArticleNo = "00103021"

LineControl.TestResult = False

LineControl.CurrentTest = "END"

LineControl.TraceID = 90016

LineControl.AdditionalText (0) = "PS250.35" "Test step number

LineControl.AdditionalText (1) = "measurement capacitor C19" "Test step text

LineControl.AdditionalText (2) = "4" "error

If LineControl.WriteCurrentStamp () = 0 Then

MsgBox "Record written"

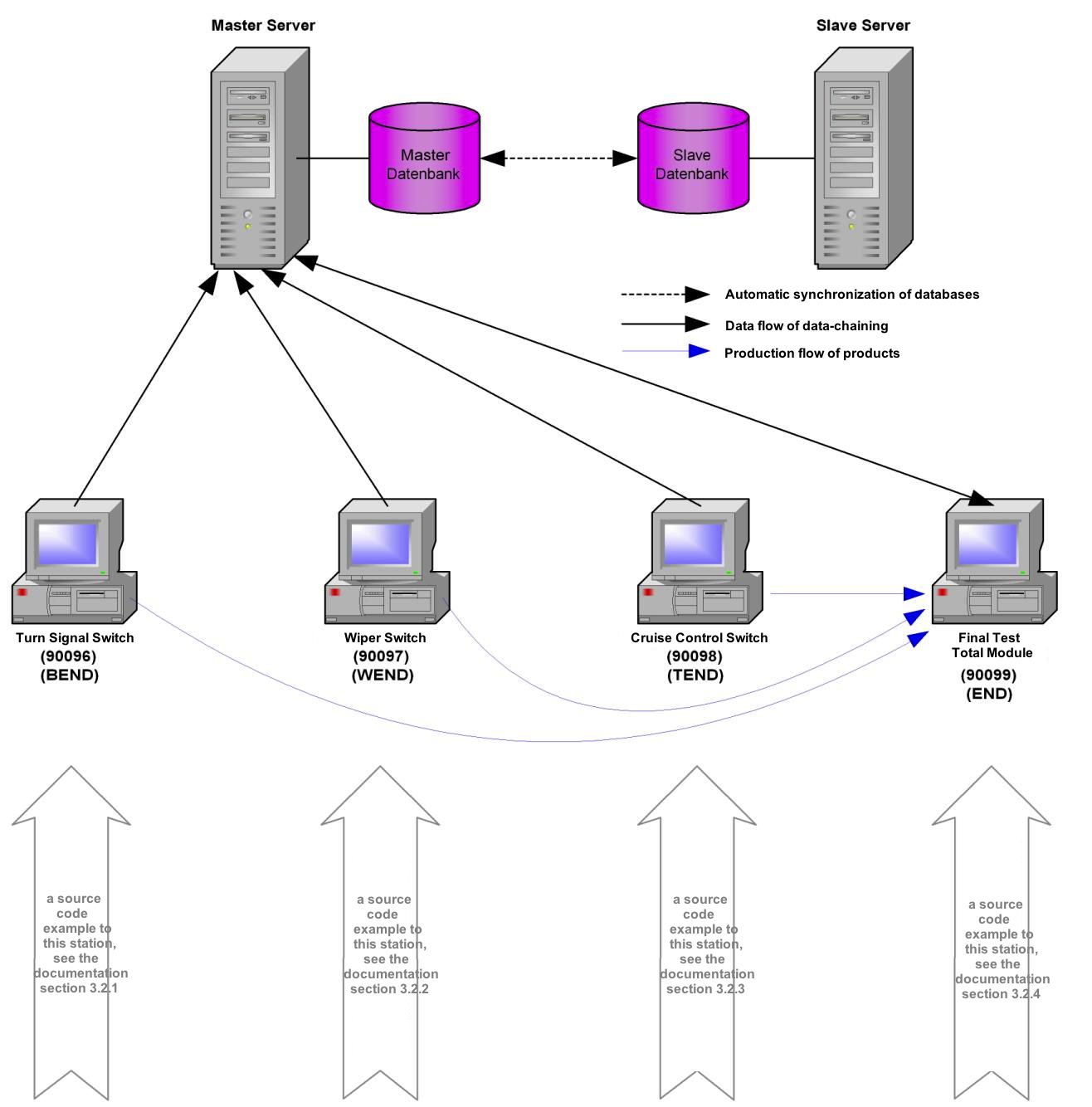
else

MsgBox "Record not written"

end if

* 1. **Chaining of multiple products**

The following diagram shows the linkage of the Line Controller 2004 ActiveX DLL of several products that flow at a given time in a new production of the entire product. To ensure this production variant, all functions of the Line Controller 2004 ActiveX DLL must be assigned the blocked parts to the total product. This is achieved that all individual products are checked with the "ReadPreviousStamp", and thereby the function parameter "PrepareForChild" is set to TRUE. If now the function "WriteCurrentStamp" is called, the DLL writes all the information from the items assigned previously to the overall product.



* + 1. **Example source code for turn signal switch final test**

LineControl.SerialNo = "0010302140230234"

LineControl.ArticleNo = "00103021"

LineControl.TestResult = True

LineControl.CurrentTest = "BEND"

LineControl.TraceID = 90096

If LineControl.WriteCurrentStamp () = 0 Then

MsgBox "Record written"

else

MsgBox "Record not written"

End If

* + 1. **Example source code forwiper switch final test**

LineControl.SerialNo = "0010310140230234"

LineControl.ArticleNo = "00103101"

LineControl.TestResult = True

LineControl.CurrentTest = "WEND"

LineControl.TraceID = 90097

If LineControl.WriteCurrentStamp () = 0 Then

MsgBox "Record written"

else

MsgBox "Record not written"

End If

* + 1. **Example source code for cruise control switch final test**

LineControl.SerialNo = "0010402140230234"

LineControl.ArticleNo = "00104021"

LineControl.TestResult = True

LineControl.CurrentTest = "TEND"

LineControl.TraceID = 90098

If LineControl.WriteCurrentStamp () = 0 Then

MsgBox "Record written"

else

MsgBox "Record not written"

End If

* + 1. **Example source code foroverall module final test**

bOK = True

LineControl.SerialNo = "0010302140230234"

LineControl.ArticleNo = "00103021"

LineControl.PreviousTest (0) = "BEND"

LineControl.SearchMode = 0

If LineControl.PreviousStamp () <> 0 Then bOK = False

LineControl.PrepareForChild ()

LineControl.SerialNo = "0010310140230234"

LineControl.ArticleNo = "00103101"

LineControl.PreviousTest (0) = "WEND"

LineControl.SearchMode = 0

If LineControl.PreviousStamp () <> 0 Then bOK = False

LineControl.PrepareForChild ()

LineControl.SerialNo = "0010402140230234"

LineControl.ArticleNo = "00104021"

LineControl.PreviousTest (0) = "TEND"

LineControl.SearchMode = 0

If LineControl.PreviousStamp () <> 0 Then bOK = False

LineControl.PrepareForChild ()

If bOK = True Then

LineControl.SerialNo = "0040302140230234"

LineControl.ArticleNo = "00403021"

LineControl.TestResult = True

LineControl.CurrentTest = "END"

LineControl.TraceID = 90099

If LineControl.WriteCurrentStamp () = 0 Then

If LineControl.MarryAllChildren () = 0 Then

MsgBox "Record written"

End If

else

MsgBox "Record not written"

End If

else

Call Kill Childs

End If

**4 Use of the component**

**4.1 Integration in a Project**

The Line Controller 2004 ActiveX DLL can be used as a component of a project, it must be made via a known link. In the following the application of a reference in the development Microsoft Visual Basic environment is explained.

**4.1.1 Creating a reference in MS Visual Basic**

Open the References dialog box by clicking in the Project menu, and select References. Here you can select from the list of available references, select "Kostal Line Controller 2004 DLL".

If this should not be available, click Browse……it will open the Add Reference dialog. This is similar to a standard dialog box for opening files. Set the file type to Execution file (\*.exe, \*.dll).Navigate to the installation directory of the Line Controller 2004 ActiveX DLL and select the file Linecontroller.dll. Click Open.

Click OK to complete the settings.

**4.1.2Using of the component under MS Visual Basic**

Once you have set the reference to the component that you can use in your Visual Basic project. First, create an instance of the component by

Private/Public LineControl As New clsDB\_Linecontrolling

About the name (line control) used here, you can access the functions and features of the Line Controller 2004 ActiveX DLL, such as

LineControl.Init

LineControl.Traceid

LineControl.MasterDB\_User

**4.2Function**

The 2004 Line Controller ActiveX DLL provides nine functions as a component of charge. These are explained in the following sections. All examples and syntax Listings refer to Microsoft Visual Basic.

**4.2.1 Init**

**Task**

The function is used to initialize the DLL. The component is thereby adapted to the requirements of the program. You check the necessary parameters to valid-speed and try to connect the specified SQL database. With successful connection, it is checked whether the necessary database as well as the necessary tables exists. If this is not the case, they are automatically created by the DLL.

**Call and parameter**

The call function is in the manner,

lReturn = LineControl.Init

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

So that the function can be properly run, at least two properties of the DLL must to be set in advance. These two parameters are described below:

INIFILE (string) specifies the configuration file to use. This contains the necessary settings. For any further information, see section 5.2 - Linecontrolling.ini. It is expected the path and the file name.

INISection (string) identifies a specific section in the configuration file, called section. Usually the name of the corresponding test set is used.

All other required properties are read from the specified configuration file and stored in properties (Property).The programmer is free to choose the properties before initialization by calling the init function. This is set by the programmer and the related values ​​are used in the configuration file. For this reason, the following is a brief overview available via the possible parameters. For more information, please refer to the individual parameter descriptions.

CurrentTest

Process

MasterDB\_Server

MasterDB\_Name

MasterDB\_User

MasterDB\_Password

SlaveDB\_Server

SlaveDB\_Name

SlaveDB\_User

SlaveDB\_Password

INIFile

INISection

KillAfterTest

LocalPath

MarkAfterTest

PreviousTest

SearchMode

ServerMode

**Error codes**

The function returns the error code of type Long Skip past navigation:

|  |  |
| --- | --- |
| 0 | No error occurred |
| -1 | IP address database is invalid |
| -2 | Missing or incorrect parameter |
| -3 | Database could not be opened |
| -4 | No previous test found |
| -5 | Missing entry in the INI file |
| -6 | Configuration file not found |
| -7 | Local path is not translated |
| -8 | Failed to initialize the logging module |
| -9 | INIFile not found |
| -10 | Invalid Server Mode |
| -11 | Unknown Error |

**4.2.2 ReadCurrentStamp**

**Task**

This function shall assume the reading of the currently written record from the database. It is intended to check whether the writing of a record has been successfully carried out or how much repeating test has passed through the product. This feature requires a working connection to the SQL database. If the server is the master-slave mode operation, the slave database is used when there is no connection to the master database. In standalone mode, the function returns an error code Skip past navigation. The function has no parameters. It is available via the control properties ArticleNo, SerialNo and CurrentTest of the DLL. For more information, please refer to the description of each property.

This function also sets a flag for the properties SerialNo, Timestamp, TraceID, Mark-After, TestResult, all 8 AdditionalFlag and all 4 AdditionalText, which ensures that if they have not already set by the programmer, before calling ReadPreviousStamp, ReadCurrentStamp, WriteCurrentStamp, MarryAllChildren, the properties listed Skip past navigation uses the values.

Example in Visual Basic:

LineControl.ArticleNo = "00101800"

LineControl.SerialNo = "12345678901234"

If LineControl.ReadCurrentStamp = 0 Then

Msgbox "Record successfully found"

else

MsgBox "Record not found"

End If

**Call**

The call function is made in the manner,

lReturn = LineControl.ReadCurrentStamp()

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

The function uses three characteristics (properties) that have been mentioned in the task description. For detailed information, please refer to the individual parameter descriptions.

**Features**

The function requires a working network connection to the SQL database.

**Return values**

The function returns the number of records, if which are found. This is not the case or an error occurs, the following error codes of type Long are given back.

|  |  |
| --- | --- |
| 0 | No error occurred |
| -1 | Entries for the item and serial number not found |
| -2 | Error while establishing a connection to the database |
| -3 | Missing or incorrect parameter |
| -4 | DLL was not initialized |
| -5 | Given to the article, serial number or the specified test, there are no records |
| -6 | Unknown error |

**4.2.3 ReadPreviousStamp**

**Task**

This function shall assume the reading of the previous records from the database that have been written by the pre-stored test system. It is intended to ensure the linking of the product. It is controlled by a plurality of parameters, in the form of characteristics (properties). They checked whether all the specified tests have been successfully carried out prior to a certain serial number and product. Furthermore, it is possible available via the Search Mode property to specify a specific search pattern. For any further information on the search patterns, please read the section in which the property search mode is described. This function requires a function-enabled connection to the SQL database. If the server is the master-slave mode operation, the slave database is used in case of connection to the master database is missing. In standalone mode, the function returns an error code Skip past navigation. The function has no parameters. It is available via the controlled properties ArticleNo, SerialNo, PreviousTest and SearchMode of Dll. For more information, please refer to the description of each property.

This function also sets a flag for the properties SerialNo, timestamp TraceID, Mark-After, TestResult, all 8 AdditionalFlag and all 4 AdditionalText, ensures that the properties listed before calling ReadPreviousStamp, ReadCurrentStamp, WriteCurrentStamp, MarryAllChildren the values back sets, where these have not already been set by the programmer.

Example in Visual Basic:

LineControl.ArticleNo = "00101800"

LineControl.SerialNo = "12345678901234"

LineControl.PreviousTest (1) = "PRE"

LineControl.SearchMode = 0

If LineControl.ReadPreviousStamp () = 0 then

Msgbox "Record successfully found"

else

MsgBox "Record not found"

end if

**Call and parameter**

The call function is made in the manner，

lReturn = LineControl.ReadPreviousStamp()

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

The function uses four characteristics (properties) that have been mentioned in the task description. For more information, please refer to the individual parameter descriptions.

**Features**

The function requires a working network connection to the SQL database.

**Return values**

The last record is mapped to the corresponding properties (TestResult, TraceID, AdditionalText, ect.).

Furthermore, the function returns the error code type Long Skip past navigation:

|  |  |  |
| --- | --- | --- |
| 0 | | No error occurred |
| -1 | | No entries for the item and serial number found |
| -2 | Failed to build the connection to the database |
| -3 | Missing or incorrect parameter |
| -4 | The part was tested on noisy pattern poorly |
| -5 | DLL was not initialized |
| -6 | To the specified item number so far there are no records |
| -7 | Unknown error |

**4.2.4 WriteCurrentStamp**

**Task**

This function shall assume the writing of records in the SQL database. This feature requires a working connection to the SQL database. If the server is the master-slave mode operation, the slave database is used when there is no connection to the master database. Synchronizing the databases is taken over by the server. As input parameters different properties are used, which are named in the following. All required properties (Properties) are shown in bold.

ArticleNo

SerialNo

TimeStamp

CurrentTest

TraceID

TestResult

AdditionalFlag(0) – AdditionalFlag(7)

AdditionalText(0) – AdditionalText(3)

MarkAfterTest

Example in Visual Basic:

LineControl.ArticleNo = "00101800"

LineControl.SerialNo = "12345678901234"

LineControl.TimeStamp = ""

LineControl.CurrentTest = "END"

LineControl.TestResult = True

LineControl.TraceID = "90058"

LineControl.AdditionaltText (0) = "test data set;-)"

If LineControl.WriteCurrentStamp () = 0 Then

Msgbox "Record successfully written"

else

Msgbox "record not written to the DB"

End If

**Call and parameter**

The call function is made in the manner，

lReturn = LineControl.ReadPreviousStamp()

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

The function uses different properties (properties) that have been mentioned in the task description. For more information, please refer to the individual property descriptions.

**Features**

The function requires a working network connection to the SQL database.

**Return values**

The function returns the error code of type Long Skip past navigation:

|  |  |
| --- | --- |
| 0 | No error occurred |
| -1 | Missing or incorrect parameter |
| -2 | DLL was not initialized |
| -3 | Record was not written |
| -4 | Unknown error |

**4.2.5 KillAllChildren**

**Task**

This function resets all read records that have been prepared as a Child Skip past navigation (see 4.2.7 PrepareForChild).This is important, for example, if a module fails in a test step and it should be retested. So that then double data records and key injuries do not occur, the kill Childs function must be called before.

**Call and parameter**

The call function is made in the manner，

lReturn = LineControl.KillAllChildren ()

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

The function has no parameters.

**Return values**

The function returns the error code of type Long Skip past navigation:

|  |  |
| --- | --- |
| 0 | No error occurred |
| -1 | DLL was not initialized |
| -2 | Unknown error |

**4.2.6 CheckDBConnection**

**Task**

This function checked the connection to the specified database. Continues to write this function in standalone mode, all linking data stored on the local hard disk to the database, if the connection has been successfully established. In the master-slave mode tries to connect to the master database, the upgrade fails, an internal bit is set so that all data is written to the slave database. When the connection is established, the bit is changed back, and the data automatically flow back into the master database. This function is called after statement of an incorrect connection of all 25 written data records, so a check is done again whether the connection to the database (master database) is available.

**Call and parameter**

The call function is made in the manner，

lReturn = LineControl.CheckDBConnection()

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

The function has no parameters.

**Return values**

The function returns the error code of type Long Skip past navigation:

|  |  |
| --- | --- |
| 0 | No error occurred |
| -1 | DLL was not initialized |
| -2 | Unknown error |

**4.2.7 PrepareForChild**

**Task**

This function can be called after (see 4.2.3 ReadPreviousStamp) the previous test of a product has been successful in carrying out a controlled manner using the ReadPreviousStamp function. After this function has been called, the current set values ​​of properties and ArticleNo and SerialNo are stored in an array. If now the MarryAllChildren(see 4.2.8 MarryAllChildren) function is called, additional information is written to the database, it makes possible at a later time an association between the previously controlled parts and the current manufacture part. For example, this approach built single switch can be identified in a module. Furthermore, a proper clean all records can be guaranteed by this store when extended storage of the data is not provided. This feature requires a working connection to the SQL database.

**Call and parameter**

The call function is made in the manner，

lReturn = LineControl.PrepareForChild()

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

The function has no parameters.

**Return values**

The function returns the error code of type Long Skip past navigation:

|  |  |
| --- | --- |
| 0 | No error occurred |
| -1 | DLL was not initialized |
| -2 | Unknown error |

**4.2.8 MarryAllChildren**

**Task**

This function combines all previously stored with the PrepareForChild (see 4.2.7 PrepareForChild) feature item and serial number of the currently selected item and writes the records to the database. For example, by this procedure, an assignment of a module, incorporated into single item is possible. Furthermore, a proper clean all records can be guaranteed by this store when a permanent storage of the data is not provided. This feature requires a working connection to the SQL database.

**Call and parameter**

The call function is made in the manner，

lReturn = LineControl.MarryAllChildren()

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

The function has no parameters.

**Return values**

The function returns the error code of type Long Skip past navigation:

|  |  |
| --- | --- |
| 0 | No error occurred |
| -1 | Record could not be written |
| -2 | DLL was not initialized |
| -3 | Unknown error |

**4.2.9 GetLastParent**

**Task**

This function searches for a part number and a serial number of the corresponding parent in the Marrying table. If multiple entries exist, the current is taken, which is assumed that older parents have been dismantled and thus no longer physically exist. This feature requires a working connection to the SQL database.

**Call and parameter**

The call function is made in the manner，

Dim bReturn As Boolean

Dim result() As String

Dim articleNumber as String

Dim serialNumber as String

articleNumber = “12345678-01”

serialNumber = “12345ABC10000”

bReturn = LineControl.GetLastParent(articleNumber, serialNumber, result)

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

**Return values**

The function returns a Boolean result, the value "true" for a record is found and the value "false" for no record is found.

A record was found, the data is written to the array, which was being furnished as the last parameter.

The evaluation of the call takes place in the manner

If bReturn = True Then

MsgBox “ElternteilArtikelnummer: “ &result(0)

MsgBox “ElternteilSeriennummer: “ &result(1)

End If

**4.2.10 GetParents**

**Task**

This function searches for a part number and a serial number, all relevant parents in the Marrying table. The output is sorted by the created timestamp of entries. This function requires a working connection to the SQL database.

**Call and parameter**

The call function is made in the manner，

Dim lRecords As Long

Dim results() As String

Dim articleNumber as String

Dim serialNumber as String

articleNumber = “12345678-01”

serialNumber = “12345ABC10000”

lRecords = LineControl.GetParents(articleNumber, serialNumber, results)

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

**Return values**

The function returns a numeric result, the value for the number of records is found. The records are written to the array, which was being furnished as the last parameter.

The evaluation of the call takes place in the manner

Dim lCounter As Long

For lCounter = 0 TolRecords – 1

MsgBoxlCounter& „. Parent Artikelnummer: „ &results(lCounter)(0)

MsgBoxlCounter& „. Parent Seriennummer: „ &results(lCounter)(1)

Next

**4.2.11 GetChildren**

**Task**

This function searches for a part number and a serial number, all relevant parts of children in the marriage table. The output is sorted by the created timestamp of entries. This function requires a working connection to the SQL database.

**Call and parameter**

The call function is made in the manner，

Dim lRecords As Long

Dim results() As String

Dim articleNumber as String

Dim serialNumber as String

articleNumber = “12345678-01”

serialNumber = “12345ABC10000”

lRecords = LineControl.GetChildren(articleNumber, serialNumber, results)

with the LineControl in 4.1.2 - is specified using the component name under MS Visual Basic.

**Return values**

The function returns a numeric result, the value for the number of records is found. The records are written to the array, which was being furnished as the last parameter.

The evaluation of the call takes place in the manner

Dim lCounter As Long

For lCounter = 0 TolRecords – 1

MsgBoxlCounter& „. Child Artikelnummer: „ &results(lCounter)(0)

MsgBoxlCounter& „. Child Seriennummer: „ &results(lCounter)(1)

Next

**4.3 Features**

The 2004 Line Controller ActiveX DLL has a total of 31 properties. These are explained in the following sections. All examples and syntax Listings refer to Microsoft Visual Basic.

**4.3.1 AdditionalFlag (index)**

**Task**

With this feature you can set 8 flags to each record and write to the database. These flags are no special meaning, so they can be freely used, thus they are dependent on the application. To put the flags property of AdditionalFlag (0) to AdditionalFlag (7) must be called.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is set to FALSE.

**Set and valid values**

The property comes with a command such as

LineControl.AdditionalFlag(0) = bValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

bValue must be a Boolean value.

**Read and Return values**

The property comes with a command such as

bValue = LineControl.AdditionalFlag(0)

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

bValue must be a Boolean value.

**Error**

The property provides no error codes.

**4.3.2 AdditionalText (index)**

**Task**

With this property, you can get four free texts to each record and write to the database. These texts expand the old function of AdditionalParameter and can be used freely. To set the text, the property of AdditionalText (0) to AdditionalText (3) must be called.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.AdditionalText(0) = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) can be any string.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.AdditionalText(0)

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is a string.

**Error**

The property provides no error codes.

**4.3.3 ArticleNo**

**Task**

The item number is written to the record in the database, or returned the selected item number. After Kostal standard, the item number must be specified with eight characters, for compatibility reasons, the indication of a longer item number is possible.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.ArticleNo = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is strings 8-30 characters.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.ArticleNo

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) will give back at least a 8-digit part numbers.

**Error**

The property provides no error codes.

**4.3.4 PreviousTest (index)**

**Task**

Sets the previous process steps that will be checked when the function is called ReadPreviousStamp. The index must be strictly specified by its specification, it is possible to control several previous continuous production steps. If the property prior to initialize the DLL is not set, it is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value "NONE".

**Set and valid values**

The property comes with a command such as

LineControl.PreviousTest(index) = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (String) is at least 3 digit strings that identify the previous test. If no test is available, the property must be that the "index" 0 can be set to "NONE".

To delete a previously set value, this must be set to "NONE". The value and all subsequent values ​​are deleted. If the PreviousTest set with index 0 to "NONE", no previous processes are expected.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.PreviousTest(index)

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned a string that identifies the previous test.

**Error**

The property provides no error codes.

**4.3.5 CurrentTest**

**Task**

Specifying the test, or return. If the property before initializing the DLL is not set then it will be set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.CurrentTest = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings between 3 and 10 characters in length.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.CurrentTest

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

Assigned sValue (string) strings describe the current, or read from the database test.

**Error**

The property provides no error codes.

**4.3.6 MasterDB\_Name**

**Task**

Specifying the database name of the master database which is to be used, in order to write the concatenation of data into the database, or return the currently set database name of the master database. If the property before initializing the DLL is not set, it is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.MasterDB\_Name = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings

**Read and Return values**

The property comes with a command such as

sValue = LineControl.MasterDB\_Name

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned a string that contains the current master database name used.

**Error**

The property provides no error codes.

**4.3.7 MasterDB\_Password**

**Task**

Specifying the database password has been the master database that is to be used to connect to the database, or the database password is currently set. If the property before initializing the DLL is not set, it is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

If this property is set again after the initialization of the DLL, it must be reinitialized.

Please only use the connection settings that you receive from the Department APB3.4.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.MasterDB\_Password = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings

**Read and Return values**

The property comes with a command such as

sValue = LineControl.DB\_Password

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned a string that contains the master database password currently used.

**Error**

The property provides no error codes.

**4.3.8 MasterDB\_Server**

**Task**

Specifying the master database of the server IP address that will be used to connect to the database, or Skip past navigation is the currently set IP address of the master database server. If the property before initializing the DLL is not set, it is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

If this property is set again after the initialization of the DLL, it must be re-initialized.

Please only use the connection settings that you receive from the Department APB3.4.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.MasterDB\_Server = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings that describe the server (eg:"193.22.212.123").

**Read and Return values**

The property comes with a command such as

sValue = LineControl.MasterDB\_Server

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) can be assigned to strings that contain the database server currently in use.

**Error**

The property provides no error codes.

**4.3.9 MasterDB\_User**

**Task**

Setting the current user for the master database to be used to connect to the database. Skip past navigation or are currently set to the user master database. If the property before initializing the DLL is not set, then this is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

If this property is set again after the initialization of the DLL, it must be re-initialized.

Please only use the connection settings that you receive from the Department APB3.4.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.MasterDB\_User = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.MasterDB\_User

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned to a string that contains the currently used user master database.

**Error**

The property provides no error codes.

**4.3.10 INIFile**

**Task**

Specifying the configuration file that will be used to initialize the line computer DLL, or returning the currently used configuration file.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has a value of App.Path + "\ Linecontrolling.ini".

**Set and valid values**

The property comes with a command such as

LineControl.INIFile = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings that contain the complete path and file name of the configuration file.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.DB\_INIFile

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) can be assigned to strings that contains the currently used configuration file.

**Error**

The property provides no error codes.

**4.3.11 INISection**

**Task**

Setting the range in the configuration file, from which all the necessary parameters to be read to ensure proper chain of the product or returning currently used section.

If this property is set again after the initialization of the DLL, it must be re-initialized.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value "SETTINGS\_TEST0".

**Set and valid values**

The property comes with a command such as

LineControl.INISection = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings that contain the complete path and file name of the configuration file.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.DB\_INISection

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) can be assigned to strings that contain the range currently used in the configuration file.

**Error**

The property provides no error codes.

**4.3.12 LocalPath**

**Task**

Determines where the concatenation of data are stored locally if the SQL database is not accessible. This setting is important only for the stand-alone mode. In the master-slave mode, the slave database will be used in the absence-like connection to the master database. If the property before initializing the DLL is not set, it is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

If this property is set again after the initialization of the DLL, it must be re-initialized.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.LocalPath = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings that contain the complete path and file name of the configuration file.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.LocalPath

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned a string that reflects the local path currently used in the DLL.

**Error**

The property provides no error codes.

**4.3.13 LogEnabled**

**Task**

Specifying whether the line computer generated DLL log entries that can make it easier for the programmer to program failures to find these.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value TRUE.

**Set and valid values**

The property comes with a command such as

LineControl.LogEnabled = bValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for bValue (Boolean) is TRUE and FALSE.

**Read and Return values**

The property comes with a command such as

bValue = LineControl.LogEnabled

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

bValue (Boolean) is assigned to TRUE or FALSE, depending on whether the logging feature is enabled or disabled.

**Error**

The property provides no error codes.

**4.3.14 LogFile**

**Task**

Specifying the file where the logging function of the line computer DLL will store the log entries.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value "Linecontrolling.log" and is located in the application directory.

**Set and valid values**

The property comes with a command such as

LineControl.LogFile = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (String) is file names including the path information.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.LogFile

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is the file name associated with the path, the log entries are currently stored in the DLL.

**Error**

The property provides no error codes.

**4.3.15 LogLevel**

**Task**

Specifying what is to record the logging functions of the line computer DLL.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has a value of 255. What are the different values ​​(LogLevel) can be set, refer to the documentation on the logging DLL.

**Set and valid values**

The property comes with a command such as

LineControl.LogLevel = lValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for lValue (Long) is 0 to 255.

**Read and Return values**

The property comes with a command such as

lValue = LineControl.LogLevel

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

lValue (Long) the current log level is assigned (0 to 255).

**Error**

The property provides no error codes.

**4.3.16 LogMaxFileSize**

**Task**

Determining how large the log file may be maximum. This is specified in bytes. If this value is value exceeded, the file is renamed and generates a second logging file.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value 1000000 (1MB).

**Set and valid values**

The property comes with a command such as

LineControl.LogMaxFileSize = lValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for lValue (Long) is 0 to 100000000.

**Read and Return values**

The property comes with a command such as

lValue = LineControl.LogMaxFileSize

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

lValue (Long) the maximum size of the logging file is allocated in bytes.

**Error**

The property provides no error codes.

**4.3.17 MarkAfterTest**

**Task**

Determining whether the concatenation of data should be marked for the selected item and serial number to delete after the test. By specifying this flag, it is no longer possible to delete unwanted data cyclically every 4 weeks. If the property before initializing the DLL is not set, it is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is set to FALSE.

**Set and valid values**

The property comes with a command such as

LineControl.MarkAfterTest = bValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for bValue (Boolean) is TRUE and FALSE.

**Read and Return values**

The property comes with a command such as

bValue = LineControl.KillAfterTest

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

bValue (Boolean) is assigned to a Boolean value.

**Error**

The property provides no error codes.

**4.3.18 SearchMode**

**Task**

Determining how the PreviousTest be combined with previous test steps to set a search pattern with which the SQL DB to search. If the property before initializing the DLL is not set, it is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

If this property is set again after the initialization of the DLL, it must be re-initialized.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value 99.

**Set and valid values**

The property comes with a command such as

LineControl.SearchMode = lValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for lValue (Long) is 0 to 3.

The meaning is:

0 = all tests must be carried out well

1 = as 0, in addition, the order must be complied with

2 = as 1, additionally need the tests have been carried out in the correct sequence one after the other

3 = as 2, plus the need to test the last test in the database to be

**Read and Return values**

The property comes with a command such as

lValue = LineControl.SearchMode

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

lValue (Long) is assigned to the current search.

**Error**

The property provides no error codes.

**4.3.19 SerialNo**

**Task**

Specifying the serial number that is to be written to the record in the database, or reading the serial number. After Kostal standard serial number with 14 digits must be specified, for compatibility reasons, the specification of a cropped-length or longer serial number is possible. At least 8 digits but must be specified.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.SerialNo = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is strings 8-30 characters.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.SerialNo

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue(string) is assigned to a string. This reflects the current read or serial number.

**Error**

The property provides no error codes.

**4.3.20 SlaveDB\_Name**

**Task**

Specifying the database name of the slave database that will be used to write the concatenation of data into the database, or returning the currently set database name of the slave database. If the property before initializing the DLL is not set, it is set in the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

If this property is set again after the initialization of the DLL, it must be re-initialized.

Please only use the connection settings that you receive from the Department APB3.4.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.SlaveDB\_Name = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.SlaveDB\_Name

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned a string containing the current used slave database name.

**Error**

The property provides no error codes.

**4.3.21 SlaveDB\_Password**

**Task**

Sets the current database password fixed the slave database that will be used to establish the connection to the database. Skip past navigation of or are currently set database password the slave database. If the property before initializing the DLL is not set, it is set in the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

If this property is set again after the initialization of the DLL, it must be re-initialized.

Please only use the connection settings that you receive from the Department APB3.4.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.SlaveDB\_Password = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.DB\_Password

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned a string that contains the slave database password currently used.

**Error**

The property provides no error codes.

**4.3.22 SlaveDB\_Server**

**Task**

Specifying the current IP address of the slave database server that will be used to establish the connection to the database, or Skip past navigation to the slave database servers currently set IP address. If the property before initializing the DLL is not set, it is set from the configuration file. If it has been previously set by hand, it remains unchanged by the Initialize in.

If this property is set again after the initialization of the DLL, it must be re-initialized.

Please only use the connection settings that you receive from the Department APB3.4.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.SlaveDB\_Server = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings that describe the IP address of the server (eg: "193.22.212.123").

**Read and Return values**

The property comes with a command such as

sValue = LineControl.SlaveDB\_Server

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) can be assigned to strings that contain the currently used IP address of the database server.

**Error**

The property provides no error codes.

**4.3.23 SlaveDB\_User**

**Task**

Setting the current user for the slave database established to be used to connect to the database, or returning the currently set user database of the slave. Is the property of not set before initializing the DLL, so this is set from the configuration file. If it has been previously set by hand, it remains unchanged by the initialization.

If this property is set again after the initialization of the DLL, it must be re-initialized.

Please only use the connection settings that you receive from the Department APB3.4.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.SlaveDB\_User = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for sValue (string) is all strings.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.SlaveDB\_User

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned to a string that contains the currently used user the slave database.

**Error**

The property provides no error codes.

**4.3.24 ServerMode**

**Task**

Setting the current server mode to be used to write the data chaining. There are two different server modes, the stand-alone mode and Master-Slave mode.

If this property is set again after the initialization of the DLL, it must be re-initialized.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value 99.

**Set and valid values**

The property comes with a command such as

LineControl.ServerMode = nValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for nValue (integer) is 0 and 1

0 = stand-alone mode

1 = master-slave mode

**Read and Return values**

The property comes with a command such as

nValue = LineControl.ServerMode

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

nValue (integer) is assigned a number that indicates the server mode currently in use.

**Error**

The property provides no error codes.

**4.3.25 TestResult**

**Task**

Determining whether the checked product is written as a good or bad part in the SQL database.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value "false".

**Set and valid values**

The property comes with a command such as

LineControl.TestResult = bValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for bValue(Boolean) are TRUE and FALSE.

**Read and Return values**

The property comes with a command such as

bValue = LineControl.TestResult

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

bValue (Boolean) is assigned a boolean value. This reflects the current read or test result.

**Error**

The property provides no error codes.

**4.3.26 TimeStamp**

**Task**

Specifying the date and time from which the information of checked product. This information is written along with all others in the database or read from it.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property is NULL.

**Set and valid values**

The property comes with a command such as

LineControl.TimeStamp = sValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

Valid valuefor sValue value (string) is a string in the format "YYYY-MM-DD hh: mm: ss"

**Read and Return values**

The property comes with a command such as

sValue = LineControl.TimeStamp

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) is assigned to a string. This reflects the current timestamp or read the record (tests).

**Error**

The property provides no error codes.

**4.3.27 TraceID**

**Task**

Setting the TraceID to be written to the record in the database, or returning the read-out TraceID. After the Kostal standard TraceID is unique and once for each machine awarded. The right to award a TraceID rests solely with the Department APB 3.4.

**Default**

If no adjustments are made ​​(also no init, see 3.2.1), the property has the value 0.

**Set and valid values**

The property comes with a command such as

LineControl.TraceID = lValue

set, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

New reading for lValue (Long) is from 0 to 99999.

**Read and Return values**

The property comes with a command such as

lValue = LineControl.TraceID

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

lValue (Long) is assigned a whole number. This reflects the current read or TraceID the test device, which has produced the record in the SQL database.

**Error**

The property provides no error codes.

**4.3.28 Version**

**Task**

Skip past navigation is the program version of the Line Controller 2004 ActiveX DLL.

**Set and valid values**

The property can not be set.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.Version

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) can be assigned to the program information in the following format:.. [major] [minor] [revision]

**Error**

The property provides no error codes.

**4.3.29 VersionInfo**

**Task**

Skip past navigation program gives information (program name, version, program description, author and copyright) Line Controller 2004 ActiveX DLL.

**Set and valid values**

The property can not be set.

**Read and Return values**

The property comes with a command such as

sValue = LineControl.VersionInfo

read, with the LineControl in 4.1.2 - use the component is under MS Visual Basic fixed name.

sValue (string) can be assigned to the program information in the following format:

[Program name] + newline +"Version" + [version number (see 4.3.20 - Version Number)] + newline +[Program Description] + newline +"Author:" + [author] + newline +[Copyright].

(All without brackets or Quotation mark.)

**Error**

The property provides no error codes.

**5. The configuration files**

The warring extension uses a configuration file, the "Linecontrolling.ini"

**5.1 Linecontrolling.ini**

In the Linecontrolling.ini the program can be configured:

|  |  |
| --- | --- |
| [General] |  |
| LogLevel=255 | Sets the log level to use the DLL |
| LogMax=1000000 | Specifies how large is the memory limit in bytes for the log file |
| LogEnabled=1 | Option to enable or disable the logging function (1 = on / 0 = off) |
| SERVERMODE=1 | Sets the fixed server mode (0 = Stand alone mode / 1 = master-slave operation) |
| UseOnlyTable=lc\_table | Specifies a fixed item table instead of searching for article numbers driven tables. If not desired, this row must be omitted or no value can be defined. |
|  |  |
| [MASTERDATABASE] |  |
| DBSERVER=193.22.194.52 | IP address of the master database server (MySQL Server) |
| DBNAME=bmw\_pl2 | Name of the master database |
| DBUSER=dbmaster | Master database user |
| DBPASS=masterdb | The user master database password |
|  |  |
| [SLAVEDATABASE] |  |
| DBSERVER=193.22.207.49 | IP address of the slave database server (MySQL Server) |
| DBNAME=bmw\_pl2 | Name of the slave database |
| DBUSER=dbslave | slave database user |
| DBPASS=slavedb | The user slave database password |
|  |  |
| [SETTINGS\_TEST] | Test area, which contains all the required information to ensure the concatenation note. |
| PREVIOUS\_TEST=ICT;FSH | Sets the previous tests. Usually at least 3 characters can be used for a test. All tests specified here must be carried out with good results. |
|  |  |
| SEARCH\_MODE=0 | Several tests are to be separated by a semicolon. If no previous test exists, or is no need to specify is "NONE".  SEARCH\_MODE = 0  Search mode for previous tests:  0 = all must be well carried out  1 = as 0 + order must be maintained  2 = as 1 + test must be run in the correct sequence one after the other  have been carried out  3 = as 2 + test must be run in the correct sequence one after the other  have been carried out, and be the last test |
| CURRENT\_TEST=PRE | Sets the current test, usually a string of at least 3 characters will also be used here. |
| MARK\_AFTER\_TEST=NO | Determines whether all process steps are to be marked after successful testing. By this mark of process steps, automatic storage, deletion or export is possible. |
| UseOnlyTable=lc\_table | Specifies a fixed inspires this Article Table section, rather than searching for article numbers driven tables. If not desired, this row must be omitted or no value can be defined. Overrides the global setting in [General] section. |

**6 Uninstall**

Remove the program using the Control Panel. Choose from the software point, the entry line controller DLL 2004 and click Remove. Since during the program execution may log files were created, the uninstaller does not always remove all of them capable of. So, delete any remaining files in the installation directory manually.

Another option is to call the setup and then select the menu item "Program removes them."