

USER'S MANUAL

Dibalscop.dll

FOR INTEGRATIONS



DIBAL

CONTENTS

1- LIBRARY DESCRIPTION.....	3
2- DATA IMPORT	3
2.1 SEND ITEMS THROUGH FILES	3
2.1.1 <i>DataSend Function</i>	3
2.1.2 <i>DataSend2 Function</i>	6
2.2 SEND ITEMS THROUGH PARAMETERS.....	7
2.2.1 <i>ItemsSend Function</i>	7
2.2.2 <i>ItemsSend2 Function</i>	8
2.3 SEND REGISTERS.....	12
2.3.1 <i>RegistersSend Function</i>	13
3- DATA EXPORT	16
3.1 SALES RECEPTION	16
3.1.1 <i>ReadRegister Function</i>	17
3.1.2 <i>CancelReadRegister Function</i>	19
4- COMMUNICATION STATUS WINDOW	20
5- EXAMPLE FOR USING THE DIBALSCOP.DLL, THROUGH PARAMETERS USING ITEMSEND FUNCTION, ITEGRATED BY C#	21

1- LIBRARY DESCRIPTION

This file is a library of communications developed in C++, which carries out basic operations in order to integrate management software with Dibal scales.

This library uses the dll "commL.dll" and "iconv.dll" to establish communication with scales.

By means of this library it is possible to import data to the scale or receive data from it. The library includes 5 accesible functions, 3 of them for Data Importation of data and 2 of them for Data Exportation.

2- DATA IMPORT

The dll has three functions used to send data to the scales:

1.- Function "DataSend" . This function allows to send articles to the scale from a file of articles and a file of scales.

2.- Function "ItemsSend" This function allows to send articles to the scales, but in this case the data are entered in the the fucntion directly by code as parameters of it.

3.- The function "RegistersSend" allows to send any type of register accepted by the DIBAL scales.

2.1 SEND ITEMS THROUGH FILES

The user must generate a file of articles named "dibalscopItems.txt" and a file of scales named "dibalscopScales.ini" in the same path of the Dibalscop.dll
The function;

```
string WINAPI DataSend (void)
```

This function searches the file of articles "dibalscopItems.txt" and the file of scales "dibalscopScales.ini" in order to send all the articles contained in this file to all the scales.

When the function completes the process, it creates a file named "dibalscopResults.txt" which shows the result of the communication process.

2.1.1 DataSend Function

Function to work with files.

This function search the item file called "dibalscopItems.txt" and the scale file "dibalscopScales.ini" in the same path that is the dll. Import all the items of the file and sent they to all the scales of the scale file.

```
string WINAPI DataSend (void);
```

Result 1: The function will return a string with the following values:

- 1) If the communication with all the Scales is correct:
Result = "OK"

- 2) If the communication in any of the scales **is not** correct: It will return the ipAddress of the scales with erroneous communication, separated with point & comma (“;”)

Result = “192.168.1.43;192.168.1.44”
- 3) If the dll (commL.dll), which is necessary for the communication, has not been added to the project, it will return a string “No commL.dll”.

Result = “No commL.dll”

Result 2: In addition, when the function have finished will created one file called “dibalscopResults.txt” with the result of the communication

- 1) If the communication with all the Scales is correct:

Result = “OK”
- 2) If the communication in any of the scales **is not** correct: It will return the ipAddress of the scales with erroneous communication, separated with point & comma (“;”)

Result = “192.168.1.43;192.168.1.44”
- 3) If the dll (commL.dll), which is necessary for the communication, has not been added to the project, it will return a string “No commL.dll”.

Result = “No commL.dll”

Note: The register generated by DataSend function to send each Item is:
 China -> **L2_C**
 Rest -> **AG (102E scale version or later is required)**

ITEMS FILE:

- File name: **dibalscopItems.txt**
- File type: **.txt**
- File format: **ANSI**
- Separator Character: ‘;’ (Character 44, 0x2C)
- Structure:

Field number	Description	Length	Type	Values
1	Identification code	6	Numeric	< 999999
2	Direct key	3	Numeric	< 999
3	Article price	6	decimal	< 9999,99
4	Name	36	Alphanumeric	
5	Type	1	Numeric	0-> wheighty 1-> unitary
6	Section	4	Numeric	<9999
7	Expiry date	10	Alphanumeric	dd/MM/yyyy -> Date ddd -> Days
8	Alterate price	1	numeric	0-> Allow 1-> Does not allow
9	Number	9	Numeric	< 999999999
10	Price factor	1	Numeric	0-> Yuan/kg 1-> Yuan/100g 2-> Yuan/500g
11	G text	1024	Alphanumeric	(Is not used)

Note: The GText at the moment is not sended to the scales.

Example:

000001,001,1.11,Item1,1,1,11/5/2012,0,10001,0

SCALES FILE:

- File name: **dibalscopScales.ini**
- File type: **.ini**
- File format: **ANSI**
- Structure:

[config]

scales -> Number of scales to communicate with.

showWindow -> Show Communications window

Values: **0** -> Don't show

1 -> Show

closeTime -> Number of seconds that the window will be show alter communication.

Values: **-1** -> Close manually

X -> Number of seconds to close automatically alter that the communication has finalized.

[scale01]

MasterAddress -> Scale master address.

IpAddress -> Scale Ip address.

TxPort -> Scale transmission port (Tx). (Is not in used)

RxPort -> Scale reception port (Rx).

Model -> Scale model. (Is not in used)

Values: **500RANGE** -> scale of gamma 500

LSERIES -> scale of L series

Display -> Scale display type. (Is not in used)

Values: **ALPHANUMERIC**: Alphanumeric display

GRAPHIC: Graphic display

Sections -> Scale associated sections (separated by commas “,”). (Is not in used)

Group -> Scale group.

LogsPath -> Logs file path. (Is not in used)

Example:

[config]

scales=2

showWindow = 1

closeTime = 2

[scale01]

MasterAddress=0

IpAddress=192.168.1.43

RxPort=3000

Model=500RANGE

[scale02]

MasterAddress=2

IpAddress=192.168.1.44

RxPort=3000

Model= 500RANGE

2.1.2 DataSend2 Function

This function permits to send to the scale the information of all the fields contained in AG register.

This function search the item file called `"dibalscopItems2.txt"` and the scale file `"dibalscopScales.ini"` in the same path that is the dll. Import all the items of the file and sent they to all the scales of the scale file.

```
string WINAPI DataSend2 (void);
```

Result 1: The function will return a string with the following values:

- 4) If the communication with all the Scales is correct:
Result = "OK"
- 5) If the communication in any of the scales **is not** correct: It will return the ipAddress of the scales with erroneous communication, separated with point & comma (",")
Result = "192.168.1.43;192.168.1.44"
- 6) If the dll (commL.dll), which is necessary for the communication, has not been added to the project, it will return a string "No commL.dll".
Result = "No commL.dll"

Result 2: In addition, when the function have finished will created one file called `"dibalscopResults.txt"` with the result of the communication

- 4) If the communication with all the Scales is correct:
Result = "OK"
- 5) If the communication in any of the scales **is not** correct: It will return the ipAddress of the scales with erroneous communication, separated with point & comma (",")
Result = "192.168.1.43;192.168.1.44"
- 6) If the dll (commL.dll), which is necessary for the communication, has not been added to the project, it will return a string "No commL.dll".
Result = "No commL.dll"

Note: The function generates the register AG in order to send the information to the scale. The register AG is supported in Range 500 from version 102E. In L series is supported 37d version.

ITEMS FILE:

- File name: `dibalscopItems.txt`
- File type: `.txt`
- File format: **ANSI**
- Separator Character: `'|'` (Character 124, 0x7C)
- Structure:

Field number	Description	Length	Type	Values
1	Action	1	Character	A , B or M (Add , Remove or Modify)
2	Identification code	6	Numeric	<= 999999
3	Direct key	3	Numeric	<= 999

4	Article price	7	decimal	<= 99999,99
5	Name	20	Alphanumeric	
6	Name2	20	Alphanumeric	
7	Type	1	Numeric	0-> wheighty 1-> unitary
8	Section	4	Numeric	<= 9999
9	Label Format	2	Numeric	<= 99
10	EAN Format	2	Numeric	<= 99
11	VAT Type	2	Numeric	00 to 05
12	Offer price	7	Decimal	<= 99999,99
13	Expiry date	10	Alphanumeric	dd/MM/yyyy -> Date ddd -> Days
14	Extra date	10	Alphanumeric	dd/MM/yyyy -> Date ddd -> Days
15	Tare	5	Decimal	<= 99,999
16	EAN Scanner	12	Alphanumeric	
17	Product class	2	Numeric	<= 99
18	Product Direct Number	2	Numeric	<= 99
19	Alterate price	1	numeric	0-> Allow 1-> Does not allow
20	G text	1024	Alphanumeric	

Example:

```
A|000001|001|1.11|Item1|Name2|0|1|21|2|1|0.8|12/8/2012|200|0.1|AACCCCCCEEEEE
E|2|45|1|Ingredient01,Ingredient02,Ingredient03
```

2.2 SEND ITEMS THROUGH PARAMETERS

It is not necessary to create a file, because all the articles and scales are inserted by code.

```
string WINAPI ItemsSend (Scale * myScales, int numScales,
                        Item * myItems, int numItems,
                        int showWindow, int closeTime)
```

This function accepts as parameters all the articles to be sent and all data of the scales. When the function is called, it sends all the articles to the scales. The function returns a string which shows the result of the communication process when it completes the process.

2.2.1 ItemsSend Function

This function allows sending a set of articles to the scales.

```
string WINAPI ItemsSend (Scale * myScales, int numScales,
                        Item * myItems, int numItems,
                        int showWindow, int closeTime)
```

Parameters:

- 1) **myScales**, Pointer to an array of "Scale" type structures with all the scales.
- 2) **numScales**, The total number of scales that the scales' array has.
- 3) **myItems**, Pointer to an array of "Item" type structures with all the articles to be sent to the scales.

- 4) **numItems**, The total number of articles that the articles' array has.
- 5) **showWindow**, Show communication window.
Values: 0 -> Don't show
 1 -> Show
- 6) **closeTime**, Number of seconds that the window will be show alter communication.
Values: -1 -> Close manually
 X -> Number of seconds to close automatically alter that the communication has finalized.

Result: The function will return a string with the following values:

- 1) If the communication with all the Scales is correct:
Result = "OK"
- 2) If the communication in any of the scales **is not** correct: It will return the ipAddress of the scales with erroneous communication, separated with point & comma (";")
Result = "192.168.1.2;192.168.1.3"
- 3) If the dll (commL.dll), which is necessary for the communication, has not been added to the project, it will return a string "No commL.dll".
Result = "No commL.dll"

Note: The register generated by `ItemsSend` function to send each Item is:
China -> **L2_C**
Rest -> **AG (102E scale version or later is required)**

2.2.2 ItemsSend2 Function

This function allows sending a set of articles to the scales.

```
string WINAPI ItemsSend2 (Scale * myScales, int numScales,
                          Item2 * myItems, int numItems,
                          int showWindow, int closeTime)
```

Parameters:

- 7) **myScales**, Pointer to an array of "Scale" type structures with all the scales.
- 8) **numScales**, The total number of scales that the scales' array has.
- 9) **myItems**, Pointer to an array of "Item2" type structures with all the articles to be sent to the scales.
- 10) **numItems**, The total number of articles that the articles' array has.
- 11) **showWindow**, Show communication window.
Values: 0 -> Don't show
 1 -> Show
- 12) **closeTime**, Number of seconds that the window will be show alter communication.
Values: -1 -> Close manually
 X -> Number of seconds to close automatically alter that the communication has finalized.

Result: The function will return a string with the following values:

- 4) If the communication with all the Scales is correct:
Result = "OK"
- 5) If the communication in any of the scales **is not** correct: It will return the ipAddress of the scales with erroneous communication, separated with point & comma (",")
Result = "192.168.1.2;192.168.1.3"
- 6) If the dll (commL.dll), which is necessary for the communication, has not been added to the project, it will return a string "No commL.dll".
Result = "No commL.dll"

Note: The function generates the register AG in order to send the information to the scale. The register AG is supported in Range 500 from version 102E. In L series is supported 37d version

SCALE STRUCTURE

This structure defines the scale with which the communication is established.

```
typedef struct _Scale
{
    int          masterAddress;
    LPSTR       ipAddress;
    int         txPort;
    int         rxPort;
    LPSTR       model;
    LPSTR       display;
    LPSTR       section;
    int         group;
    LPSTR       logsPath;
}Scale;
```

Where:

- **masterAddress:** Logic address of the scale (Master Address). The registers will be modified to assign this logic address.
Data type: `int` -> Integer without sign (4 bytes).
- **ipAddress:** IP address of the scale.
Data type: `LPSTR` -> 1 Byte Character String (Char)
- **txPort:** Port of the scale where we must connect for sending data to the scale.
Data type: `int` -> Integer without sign (4 bytes). (Is not in used)
- **rxPort:** Port of the scale where we must connect for receiving data to the scale.
Data type: `int` -> Integer without sign (4 bytes).
- **model:** Define the scale model.
Values:
`500RANGE` -> Is a Gamma 500 scale.
`LSERIES` -> Is a L series scale.
Data type: `LPSTR` -> 1 Byte Character String (Char)
- **display:** Type of scale display. (Is not in used)
Values:

ALPHANUMERIC -> Scale with alphanumeric display

GRAPHIC -> Scale with graphic display

Data type: LPSTR -> 1 Byte Character String (Char)

- **section:** Sections associated to the scale. If there are multiple sections they must be separated with commas (",").(Is not in used)
Data type: LPSTR -> 1 Byte Character String (Char)
- **group:** Group of the scale. The registers will be modified to assign this Group number.
Data type: `int` -> Integer without sign (4 bytes).
- **logsPath:** Path for the logs file, this file have all the registers of communication. If it is empty the communication logs will not be recorded. (It is not used)
Data type: LPSTR -> 1 Byte Character String (Char)

ITEM STRUCTURE

This is the structure to define the article data to be sent to the scale. It is used by the function ItemsSend.

```
typedef struct _Item
{
    int         code;
    int         directKey;
    double      price;
    LPSTR       name;
    int         type;
    int         section;
    LPSTR       expiryDate;
    int         alterPrice;
    int         number;
    int         priceFactor;
    LPSTR       textG;
}Item;
```

Where:

- **code :** Article Identification code (maxValue = 999999).
Data type: `int` -> Integer without sign (4 bytes).
- **directKey:** Direct key associated to the article (maxValue = 999).
Data type: `int` -> Integer without sign (4 bytes).
- **price:** Article price (maxValue= 9999,99). Data type: double(8 bytes)
- **name:** Article name (maxLength=36 bytes, for China).
Data type: LPWSTR -> Wide character string.
- **type:** Type of article. Data type: `int` -> Integer without sign (4 bytes).
0-> Weighed
1-> Non Weighed
- **section:** Article section. Data type: `int` -> Integer without sign (4 bytes).
- **expiryDate:** Best before date (format: dd/MM/yyyy).
Data type: LPSTR -> 1 Byte Character String (Char)

- **alterPrice:** Allows to modify the item price. Data type: `int`->Integer without sign (4 bytes).
 - 0 -> Allows to modify the price.
 - 1 -> Don't allow to modify the price.
- **number:** PLUNumber, 9 digits number to be printed in the label (maxValue=999999999)
 - Data type: `int` -> Integer without sign (4 bytes).
- **priceFactor:** Determines the weight base of the price.
 - Data type: `int` -> Integer without sign (4 bytes).
 - 0 -> Yuan/kg
 - 1 -> Yuan/100g
 - 2 -> Yuan/500g
- **textG:** Item G text. (maxLegth = 1024 bytes). Data type: LPSTR -> 1 Byte Character String (Char)
 - (It is not used)

ITEM2 STRUCTURE

This is the structure to define the article data to be sent to the scale. It is used by the function ItemsSend2

```
typedef struct _Item2
{
    char        action;
    int         code;
    int         directKey;
    double      price;
    LPSTR       name;
    LPSTR       name2;
    int         type;
    int         section;
    int         labelFormat;
    int         EAN13Format;
    int         VATType;
    double      offerPrice;
    LPSTR       expiryDate;
    LPSTR       extraDate;
    double      tare;
    LPSTR       EANScanner;
    int         productClass;
    int         productDirectNumber;
    int         alterPrice;
    LPSTR       textG;
}Item2;
```

Where:

- **action :** A (Add) , B(Remove) or M(Modify). `Char` -> 1 byte
- **code :** Article Identification code (maxValue = 999999).
 - Data type: `int` -> Integer without sign (4 bytes).
- **directKey:** Direct key associated to the article (maxValue = 999).
 - Data type: `int` -> Integer without sign (4 bytes).
- **price:** Article price (maxValue= 99999,99). Data type: double(8 bytes)
- **name:** Article name (maxLength=20 bytes).

Data type: LPSTR -> 1 Byte string.

- **Name2:** Article name2 (maxLength=20 bytes).
Data type: LPSTR -> 1 Byte string.
- **type:** Type of article. Data type: `int` -> Integer without sign (4 bytes).
0-> Weighed
1-> Non Weighed
- **section:** Article section. Data type: `int` -> Integer without sign (4 bytes).
Max Value = 9999
- **labelFormat:** Label Format. Data type: `int` -> Integer without sign (4 bytes).
Max Value = 99.
- **EANFormat:** EAN Format. Data type: `int` -> Integer without sign (4 bytes).
Max Value = 99.
- **VATType:** VAT type. Data type: `int` -> Integer without sign (4 bytes).
Max Value = 99.(The scale has five types of VAT)
- **offerPrice:** Offer price (maxValue= 99999,99). Data type: double(8 bytes)
- **expiryDate:** Best before date (format: dd/MM/yyyy).
Data type: LPSTR -> 1 Byte Character String (Char)
- **extraDate:** Extra date (format: dd/MM/yyyy).
Data type: LPSTR -> 1 Byte Character String (Char)
- **tare:** Tara (maxValue= 99,999). Data type: double(8 bytes)
- **EANScanner:** EAN Scanner (maxLength=12bytes)
Data type: LPSTR -> 1 Byte Character String (Char)
- **productClass:** Product class. Data type: `int` -> Integer without sign (4 bytes).
Max Value = 99.
- **productDirectNumber:** Product Direct Number. Data type: `int` -> Integer without sign (4 bytes). Max Value = 99.
- **alterPrice:** Allows to modify the item price. Data type: `int`->Integer without sign (4 bytes).
0 -> Allows to modify the price.
1 -> Don't allow to modify the price.
- **textG:** Item G text. (maxLegth = 1024 bytes). Data type: LPSTR -> 1 Byte Character String (Char)

2.3 SEND REGISTERS

The library allows us to send the registers with Dibal protocol to the scale.
In this way, we can fully configure the scale sending the information we need for example: articles, sales, advertising, configuration, ...

```
string WINAPI RegistersSend (Scale * myScales,  
                             int numScales,  
                             Register myRegisters[],
```

```

int numRegisters,
int showWindow,
int closeTime)

```

2.3.1 RegistersSend Function

This function allows sending a set of registers with Dibal format to the scales. See format of the registers in the file of communications registers corresponding to every scale model.

```

string WINAPI RegistersSend (Scale * myScales,int numScales,
Register myRegisters[],
int numRegisters,
int showWindow, int closeTime)

```

Parameters:

- 1) **myScales**, Pointer to an array of "Scale" type structures with all the scales.
- 2) **numScales**, The total number of scales that the scale's array has.
- 3) **myRegisters**, Pointer to an array of "Register" type structures with all the registers to be sent to the scales.
- 4) **numRegisters**, The total number of registers that the register's array has.
- 5) **showWindow**, Show communication window.
Values: 0 -> Don't show
 1 -> Show
- 6) **closeTime**, Number of seconds that the window will be show alter communication.
Values: -1 -> Close manually
 X -> Number of seconds to close automatically alter that the communication has finalized.

Result: The function will return a string with the following values:

- 1) If the communication with all the Scales is correct:
Result = "OK"
- 2) If the communication in any of the scales **is not** correct: It will return the ipAddress of the scales with erroneous communication, separated with point & comma (",")
Result = "192.168.1.2;192.168.1.3"
- 3) If the dll (commL.dll), which is necessary for the communication, has not been added to the project, it will return a string "No commL.dll".
Result = "No commL.dll"

SCALE STRUCTURE

This structure defines the scale with which the communication is established.

```

typedef struct _Scale
{
    int         masterAddress;
    LPSTR       ipAddress;
    int         txPort;
}

```

```

        int         rxPort;
        LPSTR      model;
        LPSTR      display;
        LPSTR      section;
        int         group;
        LPSTR      logsPath;
    }Scale;

```

Where:

- **masterAddress:** Logic address of the scale (Master Address). The registers will be modified to assign this logic address.
Data type: `int` -> Integer without sign (4 bytes).
- **ipAddress:** IP address of the scale.
Data type: LPSTR -> 1 Byte Character String (Char)
- **txPort:** Port of the scale where we must connect for sending data to the scale.
Data type: `int` -> Integer without sign (4 bytes). (Is not in used)
- **rxPort:** Port of the scale where we must connect for receiving data to the scale.
Data type: `int` -> Integer without sign (4 bytes).
- **model:** Define the scale model.
Values:
`500RANGE` -> Is a Gamma 500 scale.
`LSERIES` -> Is a L series scale.
Data type: LPSTR -> 1 Byte Character String (Char)
- **display:** Type of scale display. (Is not in used)
Values:
`ALPHANUMERIC` -> Scale with alphanumeric display
`GRAPHIC` -> Scale with graphic display
Data type: LPSTR -> 1 Byte Character String (Char)
- **section:** Sections associated to the scale. If there are multiple sections they must be separated with commas (",").(Is not in used)
Data type: LPSTR -> 1 Byte Character String (Char)
- **group:** Group of the scale. The registers will be modified to assign this Group number.
Data type: `int` -> Integer without sign (4 bytes).
- **logsPath:** Path for the logs file, this file have all the registers of communication. If it is empty the communication logs will not be recorded. (It is not used)
Data type: LPSTR -> 1 Byte Character String (Char)

REGISTER STRUCTURE

This structure defines the registers to send.

```

typedef struct _Register
{
    LPSTR      characters;
    int        sendCompleted;
}Register;

```

Where:

- **characters** : 128 characters string with the Dibal register format which is going to be send to the scale.

Ej: "AGM0000010010000250ITEM 1 Name 1 Name 2
10000020000000000002412110000000000025NNNNEEEEE00000000000000000000
000"

- **sendCompleted**: (It's not used)

3- DATA EXPORT

The dll includes 2 functions to receive the sales from the scales.

1.- Function "ReadRegister" This function opens the connection with the scale and reads the sales downloaded by the scale.

2.- Function "CancelReadRegister" This function allows to cancel the reception of sales started by function "ReadRegister".
The procedure for sales reception is shown below.

3.1 SALES RECEPTION

Sales reception Start continuous

- 1- Create a loop for continuous calling for function "ReadRegister"
- 2- If the function returns a 0, it means that the scale has no sales to download. Continue the loop until detect something to receive.
- 3- If the function returns a 1, it means that 1 register has been read and it has been copied in parameter 2 "registerBuffer", so this value must be received and treated.
- 4- If the function CancelReadRegister is called, then it must be entered as parameter the Handle of the socket to be canceled.
- 5- After calling this function, continue reading registers with the function ReadRegister until detecting no more information to read, that is until the reception of a 0 from the function.

Sales Reception Start and Stop

- 1- Create a loop for continuous calling for function "ReadRegister"
- 2- If the function returns a 0, it means that the scale has no sales to download. Continue the loop until detect something to receive.
- 3- If the function returns a 1, it means that 1 register has been read and it has been copied in parameter 2 "registerBuffer", so this value must be received and treated.
- 4- If the scales have configured as sales register the register HV (Menu 3.1.4), it is possible to know when the scale has no more sales to send, so, every time the register HV is received, it is necessary to check the field "Pending Messages" which is in the position 13 of the register. When this field has the value N, it indicates that there are no more sales to receive, so the sales reception can be stopped.
- 5- If the scales have configured as sales register the register LY (Menu 3.1.4).

Wait until the reception of 0 from function ReadRegister which means there are no more sales to read. At this moment, the sales reception from the scale can be stopped.

6- When the function `CancelReadRegister` is called, then it must be entered as parameter the Handle of the socket which we want to cancel.
After calling this function the reading of registers with function `ReadRegister` must continue until having no more data to read, that is until the reception of a 0 from the function.

REMARK: For a better efficiency in the sales reception, it is recommended to use the register of sales HV in the scale.

3.1.1 ReadRegister Function

This function allows to read a register from the scale, every time it is called.

To do it, first of all, enter as reference with the value 0 the parameter `serverHandle`, so when the scale is connected to the server socket of the PC, the function will return a value >0. This value will be the handle of the server socket of the PC to which the scale is connected.

When the function reads something from the socket, it returns to 1 and in the parameter 2 "register Buffer" it returns the register (sale) read coming from the scale.
Once the register has been received, repeat again the call to the function with the updated value of the handle of the socket "serverHandle" and continue reading registers.

```
int ReadRegister (int * serverHandle,
                 char * registerBuffer,
                 char * scaleIpAddress,
                 int scalePortTx,
                 char * pcIpAddress,
                 int pcPortRx,
                 int timeOut,
                 char * pathLogs)
```

Parameters:

- 1) **serverHandle**, Pointer to an integer which determines the identifier (handle) of the server socket of the PC to which the scale is connected.

Value=0 -> No connection. Nothing to read.
Value>0 -> Handle of the PC server socket to which the scale is connected.
- 2) **registerBuffer**, 130 bytes array, where the register of the scale will be received.
With every call to the function, `Dibalscop.dll` will return in this array of bytes the register read.
- 3) **scaleIpAddress**, IP address of the scale
- 4) **scalePortTx**, Transmission port of the scale (TX)
Remark: The transmission port must be different for every scale.
- 5) **pcIpAddress**, IP address of the PC (or network card from which the communication will be done).

- 6) **pcPortRx**, Reception port of the PC. The incoming connections will only be accepted at IP -> **pcIpAddress** and the port-> **pcPortRx**
Remark: Program as **pcPortRx** the transmission port of the scale-> **scalePortTx**
- 7) **timeOut**, Time out for the reception of connection request from the scale and for the reception of a register once the connection is established.
Remark: Better timeout 10 s. Never less than 10 s.
- 8) **pathLogs**, path for recording the communications logs.
Empty, no logs.
For recording logs, enter the complete path and the name of the logs file.

Answer: The function will return an integer with the following values:

0. **Waiting...Nothing to read** => The scale has no sales to download.
Timeout expired without nothing to read in the socket (without receiving registers) .
No more sales.
1. **Reading register** => Register received properly
2. **Finished** => The process of sales reception has finished properly.
- 1. **Inaccessible socket** => Error in reception, it is not possible to access to the socket status.
- 2. **Scale ends communication** => Reception of request of communication closing from the scale (END,ACK). The scale has no more sales.
- 3. **Socket is not connected** => Error in reception. Socket not connected.
- 4. **Net error** => Error in reception, Failure in network.
- 5. **Connexion error** => Error in reception. Connection failure.
- 6. **Length of register < 2** => The length of the received message is less than 2.
- 7. **Logs file error** => Logs file not open or not created or the length of the path for the logs file is too long.
- 9. **ReadRegister Error** => Error in function ReadRegister.
- 10. **Timeout without connexion** => Timeout expired without establishment of the connection.
- 11. **Connecting error** => Error connection , error when the connection has been established.
- 12. **Unexpected scale connection** => Accepted the connection with a non expected scale.
- 13. **Logs file error** => Logs file not open or not created or the length of the path for the logs file is too long.
- 14. **Scale Ip format error** => Wrong format of the IP address of the scale.
- 15. **PC Ip format error** => Wrong format of the IP address of the PC
- 19. **Open Server error** => Error in the establishment of the connection of server socket.
- 21. **Closing socket error** => Error closing socket.
- 22. **Closing socket error** => Error closing socket.
- 23. **Releasing resources error** => Error releasing resources.
- 24. **Pending registers error** => Error in test of read pending messages in the socket.
- 25. **Logs file error** => Logs file not open or not created or the length of the path for the logs file is too long.
- 29. **Close Server error** => Error closing server socket.

- 30.Cancelled => Operation properly cancelled.
- 31.FIN sending error => Error in sending of message END,ACK
- 32.Logs file error => Logs file not open or not created or the length of the path for the logs file is too long.
- 39.Canceling error => Error canceling operation

3.1.2 CancelReadRegister Function

This function allows to cancel the reception in a controlled way by sending END,ACK to the scale.

```
int CancelReadRegister(int * serverHandle,
                      char * pathLogs);
```

Parameters

- 1) **serverHandle**, Identifier (handle) of the socket from which we want to cancel the reception.
- 2) **pathLogs**, path for recording the communications logs.
Empty, no logs.
For recording logs, enter the complete path and the name of the logs file.


Answer: The function will return an integer with the following values:

- 31.Cancelled ok => Properly cancelled.
- 31.FIN sending error => Error in sending of message END,ACK
- 32.Logs file error => Logs file not open or not created or the length of the path for the logs file is too long.
- 39.Canceling error => Error canceling the operation

REMARK: After calling this function, the sales must continue being received until having no more information to read in the socket.

4- COMMUNICATION STATUS WINDOW

Dibalscop.dll shows a window containing data about the scales to communicate, the number of the registers sent, and the final state of the communication process.



Master Address	Scale IP	Tx port	Rx port	Registers	Status
00	10.1.8.43	3001	3000	12 / 12	OK
02	10.1.8.45	3001	3000	0 / 12	CONN_ERROR

“Status” column shows the final state of the communication and can contain the following messages:

- **NO CommL:** Communication has not started, because “commL.dll” file, which is necessary to communicate with scales, is not found.
- **OK:** Communication successfully completed. All registers have been sent to the scale.
- **CONN_ERROR:** Communication has not started, because the scale is not found and the connection can’t be established.
- **SEND_ERROR:** An error happened when sending registers. The connection with the scale is successfully established, but all the items could not be sent. “Registers” column shows the number of registers successfully sent.

5- Example for using the Dibalscop.dll, through parameters using ItemsSend function, itegrated by c#

Example based in DibalscopDemo 1.00A in c# using the Dibalscop.dll.

- 1) Copy the dlls Dibalscop.dll, commL.dll and iconv.dll in the path of the application where it wants to be integrated.
If you are programming, in debug or release path, and if you have finished your integration in the same path of your executable program.
- 2) Once they have been referenced, the function to be used must be imported to our Project.
Inside the class of the main form:

```
[DllImport("Dibalscop.dll")]
static extern string ItemsSend(DibalScale[] myScales,
                              int numberScales,
                              DibalItem[] myItems,
                              int numberItems
                              int showWindow, int closeTime);
```

- 3) We define and initialize one scales structure:

```
public struct DibalScale
{
    public int masterAddress;
    public string ipAddress;
    public int txPort;
    public int rxPort;
    public string model;
    public string display;
    public string section;
    public int group;
    public string logsPath;

    public DibalScale(int _masterAddress, string
                     _ipAddress, int _txPort, int _rxPort,
                     string _model, string _display,
                     string _section, int _group, string
                     _logsPath)
    {
        this.masterAddress = _masterAddress;
        this.ipAddress = _ipAddress;
        this.txPort = _txPort;
        this.rxPort = _rxPort;
        this.model = _model;
        this.display = _display;
        this.section = _section;
        this.group = _group;
        this.logsPath = _logsPath;
    }
}
```

4) We define and initialize one articles structure:

```
public struct DibalItem
{
    public int code;
    public int directKey;
    public double price;
    public string itemName;
    public int type;
    public int section;
    public string expiryDate;
    public int alterPrice;
    public int number;
    public int priceFactor;
    public string textG;

    public DibalItem(int _code, int _directKey, double
        _price, string _name, int _type,
        int _section, string _expiryDate,
        int _alterPrice, int _number, int
        _priceFactor, string _textG)
    {
        this.code = _code;
        this.directKey = _directKey;
        this.price = _price;
        this.itemName = _name;
        this.type = _type;
        this.section = _section;
        this.expiryDate = _expiryDate;
        this.alterPrice = _alterPrice;
        this.number = _number;
        this.priceFactor = _priceFactor;
        this.textG = _textG;
    }
}
```

5) We create one ArrayList to save the scales:

```
ArrayList alScale = new ArrayList();
//Default Scale variables
int scaleMasterAddressAux = 0;
string scaleIpAddressAux = string.Empty;
int scalePortTxAux = 3001;
int scalePortRxAux = 3000;
string scaleModelAux = "GAMMA500";
string scaleDisplayAux = "ALPHANUMERIC";
string scaleSectionsAux = string.Empty;
int scaleGroupAux = 0;
string scaleLogsPathAux = string.Empty;

//Create a scale
scale = new DibalScale(scaleMasterAddressAux, scaleIpAddressAux,
    scalePortTxAux, scalePortRxAux,
    scaleModelAux, scaleDisplayAux,
    scaleSectionsAux, scaleGroupAux,
    scaleLogsPathAux);

//Add a scale to ArrayList "arlScale"
arlScale.Add(scale);

//Copy the DibalScale objects of "arlScale" to an array "myScales"
myScales = (DibalScale[])arlScale.ToArray(typeof(DibalScale));
```

6) We create an ArrayList to save the articles:

```
ArrayList alItem = new ArrayList();
//Default Item variables
int itemCodeAux = 0;
int itemDirectKeyAux = 0;
double itemPriceAux = 0;
string itemNameAux = string.Empty;
int itemTypeAux = 0;
int itemSectionAux = 0;
string itemExpiryDaysAux = new string('0', 10);
int itemAlterPrice = 0;
int itemNumberAux = 0;
int itemFactorPrice = 0;
string itemTextG = string.Empty;

//Create an Item
item = new DibalItem(itemCodeAux, itemDirectKeyAux, itemPriceAux,
    itemNameAux, itemTypeAux, itemSectionAux,
    itemExpiryDaysAux, itemAlterPrice,
    itemNumberAux, itemFactorPrice, itemTextG);

//Add an item to ArrayList "arlItem"
arlItem.Add(item);

//Copy the DibalItem objects "arlItem" to a DibalItem array.
myItems = (DibalItem[])arlItem.ToArray(typeof(DibalItem));
```

7) We call the function that sends the articles to the scales.

```
string Result = string.Empty;
```

```
Result = ItemsSend(myScales, myScales.Length, myItems,  
                  myItems.Length, showWindow, closeTime);
```

8) We analyze the result of the communication:

```
if (Result == "OK")  
{  
    //Correct communication with all the scales  
}  
else if (Result == "No commL.dll")  
{  
    //we do not have the commL.dll  
}  
else  
{  
    //Some scale have not communicated  
    string[] ScalesError = Result.Split(';');  
}
```

9) We the integration program will be finished, we have to add the libraries Dibalscop.dll, commL.dll and iconv.dll in the same path of the integration executable

Astintze, 24 - Pol. Ind. Neinver - 48160 - DERIO (VIZCAYA) - SPAIN . Tel: (+34) 94 452 15 10 - Fax: (+34) 94 452 36 58

www.dibal.com

V 1.0.0.9_EN

DIBAL

The logo consists of the word "DIBAL" in a bold, sans-serif font. Below the text is a thick horizontal line that tapers at both ends, forming a stylized 'V' or wing shape.