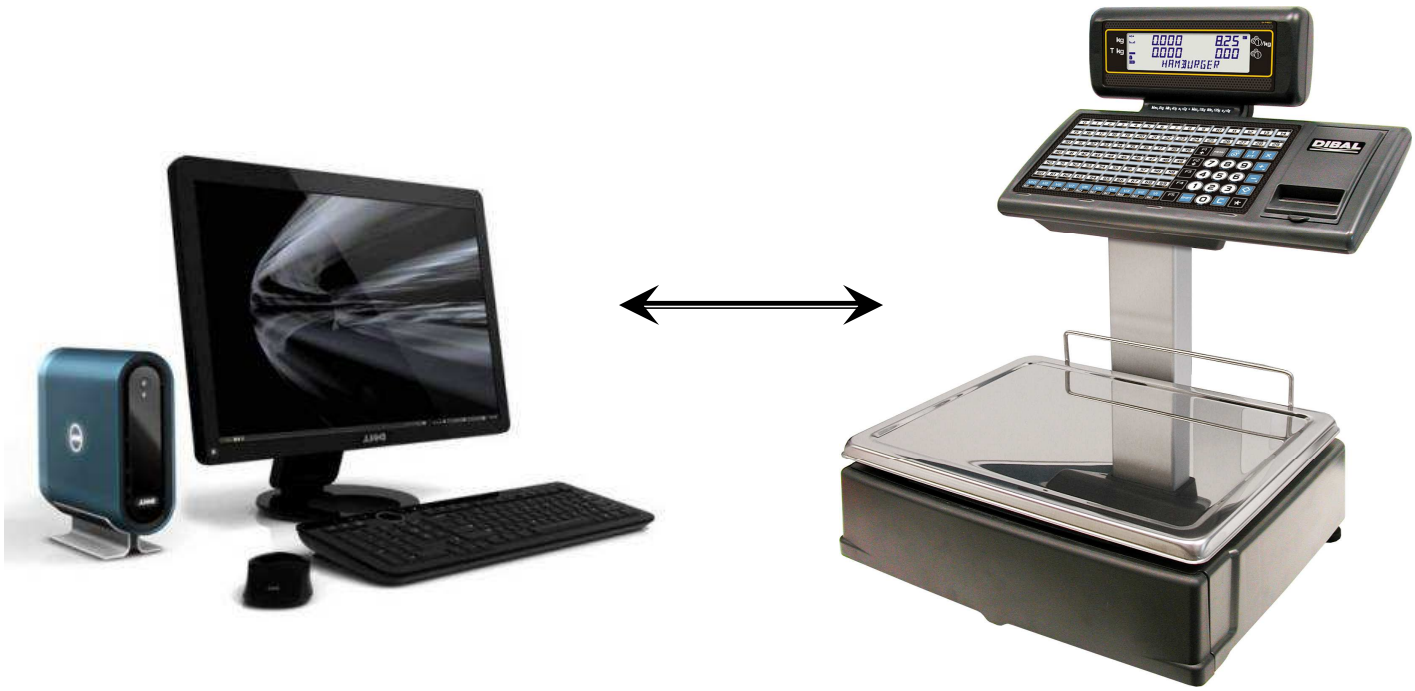
MANUAL DE USUARIO
Dibalscop.dll
PARA
INTEGRADORES



DIBAL

ÍNDICE

1- DESCRIPCIÓN DE LA LIBRERÍA

Se trata de una librería de comunicaciones construida en c++ que realiza las operaciones básicas para poder integrar programas con las balanzas de Dibal.
La librería hace uso de las dll “commL.dll” y “iconv.dll” para establecer comunicación con las balanzas.

Esta librería nos permite tanto importar datos a la balanza como recoger datos de la misma. Para ello consta de 5 funciones accesibles, 3 de ellas para Importar datos y 2 para exportar.

2- IMPORTACIÓN

La dll nos aporta 3 funciones que nos permiten enviar datos a las balanzas.
La función "DataSend" permite enviar artículos a balanzas a partir de un fichero de artículos y un fichero de balanzas
La función "ItemsSend" permite enviar artículos a balanzas, pero en este caso los datos son introducidos por código directamente en la función como parámetros de la misma.
La función "RegistersSend" permite el envío de cualquier tipo de registro soportado por las balanzas Dibal.

2.1 ENVÍO DE ARTÍCULOS MEDIANTE FICHEROS

El usuario deberá generar en el path donde este la dll "Dibalscop.dll" un fichero de artículos llamado **"dibalscopItems.txt"** y un fichero de balanzas **"dibalscopScales.ini"** .

Se utiliza la función,

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

Esta función busca el fichero de artículos "dibalscopItems.txt" y el de balanzas "dibalscopScales.ini" y enviará todos los artículos del fichero a todas las balanzas.
Cuando finaliza, generara un fichero **"dibalscopResults.txt"** con los resultados de la comunicación

2.1.1 FUNCIÓN DataSend

Función para enviar artículos contenidos en ficheros.

Esta función se encarga de buscar el fichero de artículos "**dibalscopItems.txt**" y el de balanzas "**dibalscopScales.ini**" en el path donde este instalada. Recoge todos los artículos y los envía a todas las balanzas.

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

Respuesta 1: La función devolverá un string con los siguientes valores:

1) Si todas las balanzas han comunicado correctamente.
   Respuesta = "OK"

2) Si alguna balanza **no** ha comunicado correctamente. Se devolverá la ipAddress de las balanzas que no hayan comunicado, separadas por punto y coma (";")
   Respuesta = "192.168.1.2;192.168.1.3"

3) Si no se ha añadido al proyecto la dll (commL.dll) que es necesaria para la comunicación, se devolverá el string "No commL.dll"
   Respuesta = "No commL.dll"

Respuesta 2: Además, creará el fichero "**`dibalscopResults.txt`**" con el resultado de la comunicación.

1) Si todas las balanzas han comunicado correctamente.
   Respuesta = "OK"

2) Si alguna balanza **no** ha comunicado correctamente. Se escribirá la ipAddress de las balanzas que no hayan comunicado, separadas por punto y coma (";")
   Respuesta = "192.168.1.2;192.168.1.3"

3) Si no se ha añadido al proyecto la dll (commL.dll) que es necesaria para la comunicación, se escribirá "No commL.dll"
   Respuesta = "No commL.dll"

Nota: El registro generado por la función `DataSend` para enviar cada artículo será:

China -> **L2_C**
Resto -> **AG (necesaria versión 102E de balanza o superior)**

Fichero de artículos:

- Nombre fichero: **`dibalscopItems.txt`**
- Tipo de fichero: **`.txt`**
- Formato del fichero: **`ANSI`**
- Separador: ',' Caracter 44(0x2C)
- Estructura:

Número de campo	Descripción	Longitud	Tipo	Valores
1	Código de identificación	6	Numérico	< 999999
2	Tecla directa	3	Numérico	< 999
3	Precio del artículo	6	Decimal	< 9999,99
4	Nombre	36	Alfanumérico	
5	Tipo	1	Numérico	0-> Pesado 1-> unitario
6	Sección	4	Numérico	<9999
7	Fecha de caducidad	10	Alfanumérico	dd/MM/yyyy -> Fecha ddd -> Dias
8	Alterar precio	1	Numérico	0-> Permitido 1-> No permitido
9	PLU Número	9	Numérico	< 999999999
10	Factor de precio	1	Numérico	0-> Yuan/kg 1-> Yuan/100g 2-> Yuan/500g
11	Texto G	1024	Alfanumérico	(no se usa)

Nota: El TextoG actualmente es un campo que no se envía a las balanzas.

Ejemplo:

000001,001,1.11,Item1,1,1,11/5/2012,0,10001,0
000002,002,2.22,Item2,0,2,123,0,10002,1

Fichero de balanzas:

- Nombre fichero: **`dibalscopScales.ini`**
- Tipo de fichero: **`.ini`**
- Formato del fichero: **`ANSI`**
- Estructura:

[config]
scales -> Número de balanzas con las que se va comunicar
showWindow -> Mostrar ventana de comunicaciones
Valores: `0` -> No mostrar
`1` -> Si mostrar
closeTime -> Número de segundos que se mostrará la ventana una vez finalizada la comunicación.
Valores: `-1` -> Cierre manual
`X` -> Numero de segundos para cerrar
[scale01]
MasterAddress -> Dirección maestra de la balanza
IpAddress -> Dirección Ip de la balanza
TxPort -> Puerto transmisión de la balanza. (No se usa)
RxPort -> Puerto de recepción de la balanza
Model -> Modelo de balanza. (No se usa)
Valores: `500RANGE` -> Balanza de gama 500
`LSERIES` -> Balanza de la serie L
Display -> Tipo de display. (No se usa)
Valores: `ALPHANUMERIC`: display alfanumérico
`GRAPHIC`: display gráfico
Sections -> Secciones asociadas a la balanza(separadas por comas “,”).(No se usa)
Group -> Grupo de la balanza
LogsPath -> Path del fichero de logs (No se usa)

Ejemplo:

```
[config]
scales=2
showWindow = 1
closeTime = 2

[scale01]
MasterAddress=0
IpAddress=192.168.1.10
RxPort=3000
Model=500RANGE

[scale02]
MasterAddress=2
IpAddress=192.168.1.11
RxPort=3000
Model=500RANGE
```

2.1.2 FUNCIÓN DataSend2

Función para enviar artículos contenidos en ficheros. Contiene todos los campos del registro AG.

Esta función se encarga de buscar el fichero de artículos "**`dibalscopItems2.txt`**" y el de balanzas "**`dibalscopScales.ini`**" en el path donde este instalada. Recoge todos los artículos y los envía a todas las balanzas.

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

Respuesta 1: La función devolverá un string con los siguientes valores:

4) Si todas las balanzas han comunicado correctamente.
   Respuesta = "OK"

5) Si alguna balanza **no** ha comunicado correctamente. Se devolverá la ipAddress de las balanzas que no hayan comunicado, separadas por punto y coma (";")
   Respuesta = "192.168.1.2;192.168.1.3"

6) Si no se ha añadido al proyecto la dll (commL.dll) que es necesaria para la comunicación, se devolverá el string "No commL.dll"
   Respuesta = "No commL.dll"

Respuesta 2: Además, creará el fichero "**`dibalscopResults.txt`**" con el resultado de la comunicación.

4) Si todas las balanzas han comunicado correctamente.
   Respuesta = "OK"

5) Si alguna balanza **no** ha comunicado correctamente. Se escribirá la ipAddress de las balanzas que no hayan comunicado, separadas por punto y coma (";")
   Respuesta = "192.168.1.2;192.168.1.3"

6) Si no se ha añadido al proyecto la dll (commL.dll) que es necesaria para la comunicación, se escribirá "No commL.dll"
   Respuesta = "No commL.dll"

Nota: El registro generado por la función `DataSend` para enviar cada artículo será el registro AG. Soportado en Gama 500 para versiones 102 E o superior. Para Serie L soportado en la versión 37d o superior.

Fichero de artículos:

- Nombre fichero: **`dibalscopItems2.txt`**
- Tipo de fichero: **`.txt`**
- Formato del fichero: **`ANSI`**
- Separador: '|' Caracter 124(0x7C)
- Estructura:

Número de campo	Descripción	Longitud	Tipo	Valores
1	Acción	1	Carácter	A, B o M(alta, baja o modificado)
2	Código de identificación	6	Numérico	<= 999999
3	Tecla directa	3	Numérico	<= 999
4	Precio del artículo	7	Decimal	<= 99999,99
5	Nombre	20	Alfanumérico	
6	Nombre2	20	Alfanumérico	
7	Tipo	1	Numérico	0-> Pesado 1-> unitario
8	Sección	4	Numérico	<= 9999
9	Formato de Etiqueta	2	Numérico	<= 99
10	Formato EAN 13	2	Numérico	<= 99
11	Tipo IVA	2	Numérico	00-05
12	Precio Oferta	7	Decimal	<= 99999,99
13	Fecha de caducidad	10	Alfanumérico	dd/MM/yyyy -> Fecha ddd -> Dias
14	Fecha de Extra	10	Alfanumérico	dd/MM/yyyy -> Fecha ddd -> Dias
15	Tara	5	Decimal	<= 999,99
16	EAN Scanner	12	Alfanumérico	
17	Clase producto	2	Numérico	<= 99
18	Número Rápido de Animal	2	Numérico	<= 99
19	Alterar precio	1	Numérico	0-> Permitido 1-> No permitido
20	Texto G	1024	Alfanumérico	

Ejemplo:

A|000001|001|1.11|Item1|Name2|0|1|21|2|1|0.8|12/8/2012|200|0.1|AACCCCCCEEEEEE|2|45|1|Ingrediente01,Ingrediente02,Ingrediente03

Fichero de balanzas:

- Nombre fichero: **`dibalscopScales.ini`**
- Tipo de fichero: **`.ini`**
- Formato del fichero: **`ANSI`**
- Estructura: Igual que en la función DataSend

2.2 ENVÍO DE ARTÍCULOS MEDIANTE PARÁMETROS

No es necesario generar ningún fichero ya que los artículos y las balanzas se introducen mediante código a través de la función "`ItemsSend`".

Se utiliza la función,

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

El programador deberá crear las estructuras acorde con la dll “dibalscop.dll”, una vez creadas, deberá llamar a la función “ItemsSend” y pasarle todos los datos de artículos y balanzas con las que quiera comunicar.
Cuando la función finalice, devolverá en un string el resultado de la comunicación.

2.2.1 FUNCIÓN ItemsSend

Esta función permite realizar el envío de un conjunto de artículos a la balanza.

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

Parámetros:

1) **myScales,** Puntero a un array de estructuras tipo Scale que contiene todas las balanzas con las que vamos a comunicar.

2) **numScales,** El numero total de estructuras Scale que contiene el array. El numero de balanzas con las que se va a comunicar

3) **myItems,** Puntero a un array de estructuras tipo Item que contiene todos los artículos a enviar a las balanzas.

4) **numItems,** El número total de estructuras Item que contiene el array. El número de artículos que se van a enviar.

5) **showWindow,** Mostrar ventana de comunicaciones.
   Valores: `0` -> No mostrar
   `1` -> Si mostrar

6) **closeTime,** Número de segundos que se mostrará la ventana una vez finalizada la comunicación.
   Valores: `-1` -> Cierre manual
   `X` -> Número de segundos para cerrar automáticamente.

2.2.2 FUNCIÓN ItemsSend2

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

Parameters:

1) **myScales,** Puntero a un array de estructuras tipo Scale que contiene todas las balanzas con las que vamos a comunicar.

2) **numScales,** El numero total de estructuras Scale que contiene el array. El numero de balanzas con las que se va a comunicar

3) **myItems,** Puntero a un array de estructuras tipo Item2 que contiene todos los artículos a enviar a las balanzas.

4) **numItems,** El número total de estructuras Item2 que contiene el array.
El número de artículos que se van a enviar.

5) **showWindow,** Mostrar ventana de comunicaciones.
Valores: `0` -> No mostrar
`1` -> Si mostrar

6) **closeTime,** Número de segundos que se mostrará la ventana una vez finalizada la comunicación.
Valores: `-1` -> Cierre manual
`X` -> Número de segundos para cerrar automáticamente.

Respuesta: La función devolverá un string con los siguientes valores:

1) Si todas las balanzas han comunicado correctamente.
Respuesta = "OK"

2) Si alguna balanza **no** ha comunicado correctamente. Se devolverá la ipAddress de las balanzas que no hayan comunicado, separadas por punto y coma (";")
Respuesta = "192.168.1.2;192.168.1.3"

3) Si no se ha añadido al proyecto la dll (commL.dll) que es necesaria para la comunicación, se devolvera el string "No commL.dll"
Respuesta = "No commL.dll"

Nota: El registro generado por la función `DataSend` para enviar cada artículo será el registro AG. Soportado en Gama 500 para versiones 102 E o superior. Para Serie L soportado en la versión 37d o superior.

Estructura Scale

Es la estructura que define la balanza con la que se quiere establecer la comunicación.

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

Donde:

- **masterAddress:** Dirección lógica de la balanza. Se modificarán los registros a enviar para establecerles esta dirección lógica.
Tipo dato: `int` ->Entero sin signo (4 bytes).

- **ipAddress:** Se envía la dirección IP de la balanza.
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **txPort:** El puerto de la balanza al que conectarse para enviarle registros a la balanza.
  Tipo dato: `int` ->Entero sin signo (4 bytes). (No se usa)

- **rxPort:** El puerto de la balanza al que conectarse para recibir registros a la balanza.
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **model:** Define el modelo de balanza.
  Valores:
  500RANGE -> Determina que el modelo es de la Gama 500
  LSERIES -> Determina que es una balanza de la serie L
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **display**: Tipo de display de la balanza. (No se usa)
  Valores:
  ALPHANUMERIC -> Balanza con display alfanumérico
  GRAPHIC -> Balanza con display grafico
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **section:** Secciones asociadas a la balanza. Si hay varias secciones deberán ir separadas por comas (“,”).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)
  (No se usa)

- **group:** Group al que pertenece la balanza. Se modificarán los registros a enviar para establecerles este grupo.
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **logsPath:** Ruta del fichero de logs. Si se pasa una cadena vacía, no se guardarán logs de comunicación. (No se usa)
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

Estructura Item

Es la estructura que define el artículo a enviar a la balanza para la función 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;
```

Donde:

- **code:** Código de identificación del articulo (maxValue = 999999).
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **directKey:** Tecla directa asociada al articulo (maxValue = 999).
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **price:** Precio del articulo (maxValue= 9999,99).Tipo dato: `double` (8 bytes)

- **name:** Nombre del articulo (maxLength=36 bytes, para china).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **type:** Tipo de articulo. `int` ->Entero sin signo (4 bytes).
  `0`-> Pesado.; `1`-> Unitario.

- **section:** Sección del artículo. Tipo dato: `int` ->Entero sin signo (4 bytes).

- **expiryDate:** Fecha de caducidad (formato: dd/MM/yyyy).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **alterPrice:** Permite modificar directamente el código del articulo. `int` ->Entero (4 bytes).

  `0` -> Permite modificar precio; `1` -> No permite modificar precio.

- **number:** PLUNumber, numero de 9 dígitos para imprimir en etiqueta.
  (maxValue=999999999)
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **priceFactor:** Determina la base del peso en el precio.
  Tipo dato: `int` ->Entero sin signo (4 bytes).

  `0` -> Yuan/kg; `1` -> Yuan/100g; `2` -> Yuan/500g

- **textG:** Texto G del articulo.(maxLegth = 1024 bytes).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)
  (No se usa)

Estructura Item2

Es la estructura que define el artículo a enviar a la balanza para la función 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;
```

Donde:

- **action:** A(Alta) , B(Baja) , M(Modificación). 1 byte
- **code:** Código de identificación del articulo (maxValue = 999999).
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **directKey:** Tecla directa asociada al articulo (maxValue = 999).
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **price:** Precio del articulo (maxValue= 99999,99).Tipo dato: `double` (8 bytes)

- **name:** Nombre del articulo (maxLength=20 bytes)
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **name2:** Nombre del articulo2 (maxLength=20 bytes)
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **type:** Tipo de articulo. `int` ->Entero sin signo (4 bytes).
  `0`-> Pesado.; `1`-> Unitario.

- **section:** Sección del artículo. Tipo dato: `int` ->Entero sin signo (4 bytes)
  maxValue = 9999

- **labelFormat:** Formato de etiqueta. Tipo dato: `int` ->Entero sin signo (4 bytes)
  maxValue = 99

- **EANFormat:** Formato de EAN. Tipo dato: `int` ->Entero sin signo (4 bytes)
  maxValue = 99

- **VATType:** Tipo de IVA. Tipo dato: `int` ->Entero sin signo (4 bytes)
  maxValue = 99

- **offerPrice:** Precio de oferta (maxValue= 99999,99).Tipo dato: `double` (8 bytes)

- **expiryDate:** Fecha de caducidad (formato: dd/MM/yyyy).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **extraDate:** Fecha Extra(formato: dd/MM/yyyy).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **tare:** Tara (maxValue= 99,999).Tipo dato: `double` (8 bytes)

- **EANScanner:** EAN Scanner (maxLength=12 bytes)
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **productClass:** Clase de producto (maxValue = 99).
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **productDirectNumber:** Número rápido de producto (maxValue = 99).
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **alterPrice:** Permite modificar directamente el código del articulo. `int` ->Entero (4 bytes).

  `0` -> Permite modificar precio; `1` -> No permite modificar precio.

- **textG:** Texto G del articulo.(maxLegth = 1024 bytes).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

2.3 ENVÍO DE REGISTROS

La librería nos permite enviar registros con protocolo Dibal a la balanza. De este modo podemos configurar completamente la balanza enviando la información que necesitemos por ej: artículos, vendedores, publicidad, configuración,…

Se utiliza la función;

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

2.3.1 FUNCIÓN RegistersSend

Esta función permite realizar el envío de un conjunto de registros con formato Dibal a la balanza.
Ver formato de registros en BD de registros de dibal ->Communication Registers.mdb

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

Parameters:

1) **`myScales,`** Puntero a un array de estructuras tipo Scale que contiene todas las balanzas con las que vamos a comunicar.

2) **`numScales,`** El numero total de estructuras Scale que contiene el array.
   El numero de balanzas con las que se va a comunicar

3) **`myRegisters,`** Puntero a un array de estructuras tipo Register que contiene todos los registros a enviar a las balanzas.

4) **`numRegisters,`** El número total de estructuras Register que contiene el array.
   El número de registros que se van a enviar.

5) **showWindow,** Mostrar ventana de comunicaciones.
Valores: 0 -> No mostrar
1 -> Si mostrar

6) **closeTime,** Número de segundos que se mostrará la ventana una vez finalizada la comunicación.
Valores: -1 -> Cierre manual
X -> Número de segundos para cerrar automáticamente.

Respuesta: La función devolverá un string con los siguientes valores:

1) Si todas las balanzas han comunicado correctamente.
Respuesta = "OK"

2) Si alguna balanza **no** ha comunicado correctamente. Se devolverá la ipAddress de las balanzas que no hayan comunicado, separadas por punto y coma (";")
Respuesta = "192.168.1.2;192.168.1.3"

3) Si no se ha añadido al proyecto la dll (commL.dll) que es necesaria para la comunicación, se devolvera el string "No commL.dll"
Respuesta = "No commL.dll"

Estructura Scale

Es la estructura que define la balanza con la que se quiere establecer la comunicación.

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

Donde:

- **masterAddress:** Dirección lógica de la balanza. Se modificarán los registros a enviar para establecerles esta dirección lógica.
Tipo dato: int ->Entero sin signo (4 bytes).

- **ipAddress:** Se envía la dirección IP de la balanza.
Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **txPort:** El puerto de la balanza al que conectarse para enviarle registros a la balanza.
Tipo dato: int ->Entero sin signo (4 bytes). (No se usa)

- **rxPort:** El puerto de la balanza al que conectarse para recibir registros a la balanza.

Tipo dato: `int` ->Entero sin signo (4 bytes).

- **model:** Define el modelo de balanza. (No se usa)
  Valores:
  500RANGE -> Determina que el modelo es de la Gama 500
  LSERIES -> Determina que es una balanza de la serie L
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **display**: Tipo de display de la balanza. (No se usa)
  Valores:
  ALPHANUMERIC -> Balanza con display alfanumérico
  GRAPHIC -> Balanza con display grafico
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **section:** Secciones asociadas a la balanza. Si hay varias secciones deberán ir separadas por comas (“,”).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)
  (No se usa)

- **group:** Group al que pertenece la balanza. Se modificarán los registros a enviar para establecerles este grupo.
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **logsPath:** Ruta del fichero de logs. Si se pasa una cadena vacía, no se guardarán logs de comunicación. (No se usa)
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

Estructura Register

Esta estructura define los registros a enviar

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

Donde:

- **`characters`:** String de 128 caracteres que forma el registro de formato Dibal que se enviará a la balanza.
  LPSTR -> Cadena de caracteres de 1 byte. (Char)

  Ej: “AGM0000010010000250ITEM 1 Name 1 Name 2 10000020000000000024121100000000000025NNNNEEEEE00000000000000000000000000”

- **`sendCompleted`:** (No se usa)

2.4 ENVÍO DE IMÁGENES

Dibalscop permite el envío de registros de imágenes, obtenidos mediante la función ImageRegisterGenerator de la librería GraphicImage.dll, a través de dos métodos o funciones:

- ImageSend: Permite enviar los registros de una imagen a una única balanza.
- MultiImageSend: Permite enviar los registros de varias imágenes a varias balanzas.

2.4.1 FUNCIÓN ImageSend

```
string WINAPI ImageSend(Scale myScale, Image myImage, int
iShowWindow, int iCloseTime)
```

Parámetros:

1) **myScale,** estructura de tipo Scale que contiene la información de la balanza a la que se debe realizar la comunicación.

2) **myImage,** estructura de tipo Image que contiene los datos de la imagen convertidos en registros para ser enviados.

3) **showWindow,** Mostrar ventana de comunicaciones.
   Valores: `0` -> No mostrar
   `1` -> Si mostrar

4) **closeTime,** Número de segundos que se mostrará la ventana una vez finalizada la comunicación.
   Valores: `-1` -> Cierre manual
   `X` -> Número de segundos para cerrar automáticamente.

Respuesta: La función devolverá un string con los siguientes valores:

4) Si todas las balanzas han comunicado correctamente.
   Respuesta = "OK"

5) Si alguna balanza **no** ha comunicado correctamente. Se devolverá la ipAddress de las balanzas que no hayan comunicado, separadas por punto y coma (";")
   Respuesta = "192.168.1.2;192.168.1.3"

6) Si no se ha añadido al proyecto la dll (commL.dll) que es necesaria para la comunicación, se devolvera el string "No commL.dll"
   Respuesta = "No commL.dll"

2.4.2 FUNCIÓN MultiImageSend

```
string WINAPI MultiImageSend(Scale * myScales, int numScales,
Image * myImages, int numImages, int iShowWindow, int
iCloseTime)
```

Parámetros:

1) **myScales,** Puntero a un array de estructuras tipo Scale que contiene todas las balanzas con las que vamos a comunicar.

2) **numScales,** El numero total de estructuras Scale que contiene el array.
El numero de balanzas con las que se va a comunicar

3) **myImages,** Puntero a un array de estructuras tipo Images que contiene todos los registros de imágenes a enviar a las balanzas.

4) **numImages,** El número total de estructuras Images que contiene el array.

5) **showWindow,** Mostrar ventana de comunicaciones.
Valores: `0` -> No mostrar
`1` -> Si mostrar

6) **closeTime,** Número de segundos que se mostrará la ventana una vez finalizada la comunicación.
Valores: `-1` -> Cierre manual
`X` -> Número de segundos para cerrar automáticamente.

Respuesta: La función devolverá un string con los siguientes valores:

7) Si todas las balanzas han comunicado correctamente.
Respuesta = "OK"

8) Si alguna balanza **no** ha comunicado correctamente. Se devolverá la ipAddress de las balanzas que no hayan comunicado, separadas por punto y coma (";")
Respuesta = "192.168.1.2;192.168.1.3"

9) Si no se ha añadido al proyecto la dll (commL.dll) que es necesaria para la comunicación, se devolvera el string "No commL.dll"
Respuesta = "No commL.dll"

Estructura Scale

Es la estructura que define la balanza con la que se quiere establecer la comunicación.

```
typedef struct _Scale
{
    int         masterAddress;
    LPSTR       ipAddress;
    int         txPort;
    int         rxPort;
    LPSTR       model;
    LPSTR       display;
```

```
        LPSTR       section;
        int         group;
        LPSTR       logsPath;
}Scale;
```

Donde:

- **masterAddress:** Dirección lógica de la balanza. Se modificarán los registros a enviar para establecerles esta dirección lógica.
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **ipAddress:** Se envía la dirección IP de la balanza.
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **txPort:** El puerto de la balanza al que conectarse para enviarle registros a la balanza.
  Tipo dato: `int` ->Entero sin signo (4 bytes). (No se usa)

- **rxPort:** El puerto de la balanza al que conectarse para recibir registros a la balanza.
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **model:** Define el modelo de balanza. (No se usa)
  Valores:
  500RANGE -> Determina que el modelo es de la Gama 500
  LSERIES -> Determina que es una balanza de la serie L
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **display**: Tipo de display de la balanza. (No se usa)
  Valores:
  ALPHANUMERIC -> Balanza con display alfanumérico
  GRAPHIC -> Balanza con display grafico
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

- **section:** Secciones asociadas a la balanza. Si hay varias secciones deberán ir separadas por comas (“,”).
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)
  (No se usa)

- **group:** Group al que pertenece la balanza. Se modificarán los registros a enviar para establecerles este grupo.
  Tipo dato: `int` ->Entero sin signo (4 bytes).

- **logsPath:** Ruta del fichero de logs. Si se pasa una cadena vacía, no se guardarán logs de comunicación. (No se usa)
  Tipo dato: LPSTR -> Cadena de caracteres de 1 byte. (Char)

Estructura Image

Una estructura de este tipo es retornada por la función *ImageRegisterGenerator* de la l librería *GraphicImage.dll,* la cual procesa y convierte imágenes en registros que son reconocibles por las balanzas. La estructura se define de la siguiente manera:

```
typedef struct _Image
{
        LPSTR       bitmap;
        int         len;
}Image;
```

Donde:

- **bitmap:** Es un string que contiene concatenados todos los registros de una imagen.
- **len:** Indica la longitud del string.

2.4.3 FUNCIÓN ImageRegisterGenerator(GraphicImage.dll)

Esta function no pertenece a la librería Dibalscop.dll sino que está contenida en la librería GraphicImage.dll desarrollada en C#. Se debe agregar como referencia al proyecto. Convierte una imagen en registros para ser enviados a la balanza a través de las funciones de envío de imágenes de Dibalscop.

ImageRegisterGenerator es un método que forma parte de la clase ImagePalletized.

```
public static ImageStruct ImageRegisterGenerator(int type, int
index, string pathImage, int assignPlu, int displaySize)
```

Parámetros:

1) **`type,`** Tipo de imagen. 1-> Publicidad, 2-> Artículo. Variable int de 4 bytes.

2) **`index,`** El número de orden de la imagen en la máquina. Variable int de 4 bytes

3) **`pathImage, Ubicación y nombre de la imagen a procesar.`**

4) **`assignPlu,`** Para determinar si se asignará la imagen al artículo cuyo código es igual al campo index. En caso de no existir el artículo se dará de alta.

5) **`displaySize,`** determina el tamaño del display de la balanza a la que se le va a enviar la imagen

Respuesta: La función retorna una estructura llamada ImageStruct.

Estructura ImageStruct

Es una estructura que forma parte de la clase ImagePalettized y está compuesta de los siguientes campos:

```
public struct ImageStruct
{
      public string registers;
      public int len;
}
```

`Donde:`

- **registers:** Es un string que contiene concatenados todos los registros del imagen a enviar
- **len**: Define la longitud del string.

Esta estructura es lo que se pasará como parámetro a las funciones de envío de imágenes de Dibalscop.dll

2.4.4 FUNCIÓN ImageFileGenerator(GraphicImage.dll)

Esta función no pertenece a la librería Dibalscop.dll sino que está contenida en la librería GraphicImage.dll desarrollada en C#. Se debe agregar como referencia al proyecto. Crea un fichero “txt” con el resultado de convertir una imagen en registros en una ruta determinada.

ImageFileGenerator es un método que forma parte de la clase ImagePalletized.

```
public static bool ImageFileGenerator(int masterAddress, int
type, int orden, string pathImage, string pathTxResul,int
fileCat, int assignPlu, int displaySize)
```

Parámetros:

1) **masterAddress,** Dirección lógica de la balanza. Se modificarán los registros a enviar para establecerle esta dirección lógica. Variable int de 4 bytes.

2) **type,** Tipo de imagen. 1-> Publicidad, 2-> Artículo. Variable int de 4 bytes.

3) **index,** El número de orden de la imagen en la máquina. Variable int de 4 bytes

4) **pathImage,** Ubicación y nombre de la imagen a procesar.

5) **pathTxResul,** Ubicación y nombre donde se va a guardar el txt generado.

6) **fileCat,** Sobrescribe o concatena el txt. 0-> Sobrescribe, 1-> Concatena.

7) **assignPlu,** Para determinar si se asignará la imagen al artículo cuyo código es igual al campo index. En caso de no existir el artículo se dará de alta.

8) **displaySize,** determina el tamaño del display de la balanza a la que se le va a enviar la imagen

Respuesta: La función retorna una bool que indica si se ha generado el txt. True-> Se ha generado, False-> No se ha generado.

3- EXPORTACIÓN

La dll nos aporta 2 funciones para poder recoger las ventas de las balanzas.
La función "ReadRegister" se encarga de abrir la conexión con la balanza y leer las ventas que la balanza descarga.
La función "CancelReadRegister" permite cancelar la recogida de ventas iniciada por la función "ReadRegister".
El proceso de recogida de ventas se realiza de la siguiente manera:

3.1 RECOGIDA DE VENTAS

Recogida de ventas Start continuous

1- Crear bucle que llame continuamente a la función "ReadRegister"
2- Si la función devuelve 0, quiere decir que la balanza no tiene nada que descargar. Seguiremos en el bucle hasta que haya algo que recibir.
3- Si devuelve 1 quiere decir que ha leído un registro y lo ha copiado en el parámetro 2 "registerBuffer", por lo que deberemos recoger ese valor y tratarlo.
4- Si se llama a la función CancelReadRegister se le tendrá que pasar como parámetro el Handle del socket que queremos cancelar. Después de haber llamado a esta función deberemos seguir leyendo registros con la función ReadRegister hasta que no haya mas que leer, es decir, hasta que la función devuelva 0.

Recogida de ventas Start and Stop

1- Crear bucle que llame continuamente a la función "ReadRegister"
2- Si la función devuelve 0, quiere decir que la balanza no tiene nada que descargar. Seguiremos en el bucle hasta que haya algo que recibir.
3- Si devuelve 1 quiere decir que ha leído un registro y lo ha copiado en el parámetro 2 "registerBuffer", por lo que deberemos recoger ese valor y tratarlo.
4- Si las balanzas tienen configurado como registro de ventas (menú 3.1.4) el registro HV, podremos saber cuando la balanza ya no tiene mas ventas que soltar, para ello cada vez que recibimos un registro HV, miramos en el campo "mensajes pendientes" que esta en la posición 13 del registro y si viene una N indicará que no hay mas ventas que recoger por lo que podremos detener la recogida de ventas.
5- Si las balanzas tienen configurado como registro de ventas (menú 3.1.4) el registro LY, Deberemos esperar a que la función ReadRegister nos devuelva un 0 indicando que no hay mas ventas que leer. En este momento podremos dar por finalizada la recogida de ventas para esa balanza.
6- Si se llama a la función CancelReadRegister se le tendrá que pasar como parámetro el Handle del socket que queremos cancelar. Después de haber llamado a esta función deberemos seguir leyendo registros con la función ReadRegister hasta que no haya mas que leer, es decir, hasta que la función devuelva 0.

NOTA: Se aconseja para una recogida de ventas mas eficiente, el uso del registro de ventas HV por parte de la balanza.

3.1.1 FUNCIÓN ReadRegister

Esta función permite leer un registro de la balanza cada vez que es llamada.
Para ello inicialmente habrá que pasarle el parámetro 1 "`serverHandle`" por referencia con el valor igual a 0, de manera que cuando la balanza se conecte al socket servidor del PC la función nos devolverá este parámetro con un valor > 0. Este valor será el handle del socket servidor PC al cual se ha conectado la balanza.
Cuando la función lee algo del socket retorna un 1, y en el parámetro 2 "`registerBuffer`" nos devuelve el registro (venta) leído procedente de la balanza.
Una vez recogido el registro, volvemos a repetir la llamada a la función con el valor del haldle del socket actualizado "`serverHandle`" y continuamos leyendo registros.

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

Parámetros:

1) **`serverHandle,`** Puntero a un entero que determina el identificador (handle) del socket servidor del PC al cual se conecta la balanza.

   `Valor=0 ->` No hay conexión. No hay nada que leer.
   `Valor>0 ->` Handle del socket servidor del PC al que se ha conectado la balanza.

2) **`registerBuffer,`** Array de 130 bytes donde se recibirá el registro de la balanza.
   Con cada llamada a la funcion, Dibalscop.dll nos devolverá en este array de bytes el registro leido.

3) **`scaleIpAddress,`** Dirección IP de la balanza.

4) **`scalePortTx,`** Puerto de transmision de la balanza (TX)
   `Nota:` los puertos de envío de la balanza deberán ser diferentes para cada balanza.

5) **`pcIpAddress,`** Dirección IP del PC o tarjeta de red desde la que se quiere comunicar.

6) **`pcPortRx,`** Puerto de recepción del PC. Solo se aceptarán conexiones entrantes por la IP -> **`pcIpAddress`** y el puerto-> **`pcPortRx`**
   Nota: Poner como **`pcPortRx`** el puerto de transmisión de la balanza -> **`scalePortTx`**

7) **`timeOut,`** Tiempo de espera máximo para recibir petición de conexión por parte de la balanza, asi como para recibir un registro una vez que la conexion esta establecida.
   Nota: Timeout óptimo 10s**.** No bajar nunca de 10 segundos.

8) **pathLogs,** path pdonde se escribirán los logs de la comunicación.
Si está vacio, no se escribirán logs.
Si no rellenar con el path completo y el nombre del fichero de logs.

Respuesta: La función devolverá un entero con los siguientes valores:

`0.Waiting...Nothing to read =>` La balanza no tiene nada que descargar. Timeout agotado sin nada que leer en el socket (sin recibir registros). No hay mas ventas.
`1.Reading register =>` El registros se ha recibido correctamente.
`2.Finished =>` El proceso de recogida de ventas ha finalizado correctamente.
`-1.Inaccessible socket =>` Error en la recepcion, no se puede acceder al estado del socket.
`-2.Scale ends communication =>` Se ha recibido una peticion de cierre por parte de la balanza(FIN,ACK). La balanza no tiene mas ventas.
`-3.Socket is not connected =>`Error en recepcion, el socket no esta conectado.
`-4.Net error =>` Error en la recepcion, fallo en la red.
`-5.Connexion error =>` Error en la recepcion, fallo de conexion.
`-6.Length of register < 2 =>` La longitud del mensaje recibido es menor que 2.
`-7.Logs file error =>` No se ha podido abrir o crear el fichero de logs o la ruta del fichero de logs es demasiado larga.
`-9.ReadRegister Error =>` Error en la funcion ReadRegister.
`-10.Timeout without connexion =>` Timeout Timeout agotado sin poder establecer la conexion.
`-11.Connecting error =>` Error conexion, se ha producido un error al intentar establecer la conexion.
`-12.Unexpected scale connection =>` Se ha aceptado una conexion con una balanza que no es la que se esperaba.
`-13.Logs file error =>` No se ha podido abrir o crear el fichero de logs o la ruta del fichero de logs es demasiado larga.
`-14.Scale Ip format error =>` La dirección IP de la balanza tiene un formato incorrecto.
`-15.PC Ip format error =>` La dirección IP del PC tiene un formato que es incorrecto.
`-19.Open Server error =>` Error al establecer la conexión del socket servidor.
`-21.Closing socket error =>` Error al cerrar el socket.
`-22.Closing socket error =>` Error al cerrar el socket.
`-23.Releasing resources error =>` Error al liberar recursos.
`-24.Pending registers error =>` Error al comprobar si hay mensajes pendietes de lectura en el socket.
`-25.Logs file error =>` No se ha podido abrir o crear el fichero de logs o la ruta del fichero de logs es demasiado larga.
`-29.Close Server error =>` Error al Cerrar el socket servidor.
`30.Cancelled =>` Operación cancelada correctamente.
`-31.FIN sending error =>` Error en el envío del mensaje FIN,ACK
`-32.Logs file error =>` No se ha podido abrir o crear el fichero de logs o la ruta del fichero de logs es demasiado larga.
`-39.Canceling error =>` Error al cancelar operación.

3.1.2 FUNCIÓN CancelReadRegister

Esta función permite cancelar de forma controlada la recepción de ventas enviando un FIN,ACK a la balanza.

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

Parámetros:

1) **serverHandle,** Identificador (handle) del socket del cual queremos cancelar la recepción.
2) **pathLogs,** path donde se escribirán los logs de la comunicación.
   Si está vacío, no se escribirán logs.
   Si no rellenar con el path completo y el nombre del fichero de logs.

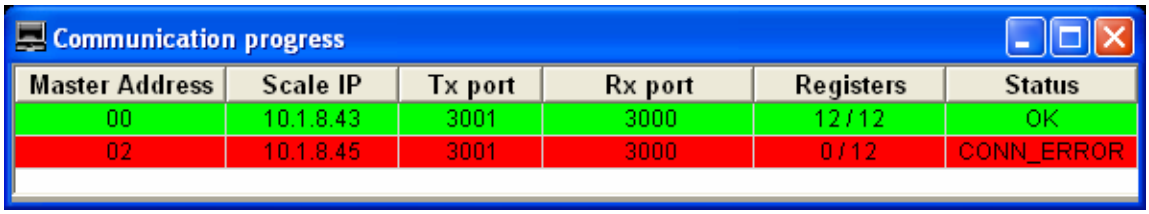
Respuesta: La función devolverá un entero con los siguientes valores:

31.Cancelled ok => Cancelado correctamente.
-31.FIN sending error => Error en el envío del mensaje FIN,ACK
-32.Logs file error => No se ha podido abrir o crear el fichero de logs o la ruta del fichero de logs es demasiado larga.
-39.Canceling error => Error al cancelar operación.

NOTA: Despues llamar a esta funcion debemos seguir recogiendo ventas hasta que no haya nada mas que leer en el socket.

4- VENTANA DE COMUNICACIONES

Dibalscop.dll puede mostrar una ventana de comunicaciones con los datos de la comunicación, Dirección de maestra, Dirección Ip, puerto Tx, Puerto Rx, Modelo, Número de registros enviados/Total, Status de la comunicación.



"Status" nos dice como ha finalizado la comunicación, y puede mostrar los siguientes valores:

- **NO CommL**: La comunicación no ha empezado, por que no se ha encontrado la dll "commL.dll" necesaria para la comunicación.
- **OK**: Comunicación completada satisfactoriamente. Todos los registros han sido enviados a la balanza.
- **CONN_ERROR**: No se ha podido conectar a la balanza. Revise la conexión.
- **SEND_ERROR**: Error durante la comunicación. La conexión se ha establecido pero todos los registros no han sido enviados. La columna registros, muestra el número de registros enviados.

5- Ejemplo de uso de Dibalscop.dll mediante la función ItemsSend, integrada en C#

Ejemplo basado en DibalscopDemo 1.00A de integración en c# de la dll Dibalscop.dll

1) Copiar las dlls Dibalscop.dll, commL.dll y "iconv.dll" en el path de la aplicación con la cual queremos integrar.
   Si estás programando, deberás copiar estas librerías en debug o en release.
   Una vez acabada la integración, habrá que copiar estas librerías en el path donde este el ejecutable de la integración.

2) Una vez que hemos copiado las librerías, deberemos importar la función que envía los artículos a nuestro proyecto. Dentro de la clase del formulario principal:

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

3) Definimos la estructura de la balanza:

```
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) Definimos la estructura del artículo:

```
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) Creamos un array list para añadir las balanzas:

```
ArrayList alScale = new ArrayList();
//Default Scale variables
int scaleMasterAddressAux = 0;
string scaleIpAddressAux = string.Empty;
int scalePortTxAux = 3001;
int scalePortRxAux = 3000;
string scaleModelAux = "500RANGE";
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) Creamos un array list para añadir los artículos:

```
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) Llamamos a la función que envía los artículos a las balanzas.

```
string Result = string.Empty;

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

8) Analizamos el resultado de la comunicación:

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

9) Una vez finalizada la integración, deberemos copiar las librerías Dibalscop.dll , commL.dll y “iconv.dll” en el mismo path donde instalemos el ejecutable de la integración.