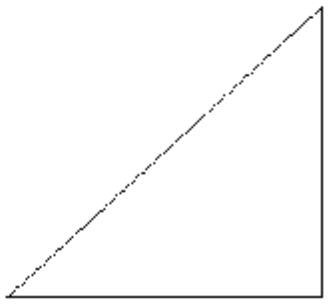
XGatDll.Dll

Communications Bookshop for MÁXIMA scales



Grupo Campesa
Balanzas y básculas

12/11/03

TABLE OF CONTENTS

1) TOPOLOGY OF MAXIMA SCALES NETWORK

There are basically 3 kinds of Local Area Network (LAN) topology for networking the Máxima Range weighing scales and connecting them to computers for local access:

Ethernet Network

RS485 Network

485/Ethernet Network

ETHERNET NETWORK

Also grouped under the Ethernet network are three basic types of Local Area Network (LAN) topologies, the most common ones, however, are the “Star” and “Bus” types; the “Bus” type has fallen into disuse and we shall thus be solely explaining the “Star” type.

The most extensively found cabling type is **10Base-T, UTP Cat 5 type**

Ehernet segun el cable usado

Tipo	Medio	Ancho de banda máximo	Longitud máxima de segmento	Topología Física	Topología Lógica
10Base5	Coaxial grueso	10 Mbps	500 m	Bus	Bus
10Base-T	UTP Cat 5	10 Mbps	100 m	Estrella; Estrella Extendida	Bus
10Base-FL	Fibra óptica multimodo	10 Mbps	2.000 m	Estrella	Bus
100Base-TX	UTP Cat 5	100 Mbps	100 m	Estrella	Bus
100Base-FX	Fibra óptica multimodo	100 Mbps	2.000 m	Estrella	Bus
1000Base-T	UTP Cat 5	1000 Mbps	100 m	Estrella	Bus

Specifications of Ethernet 10Base-T UTP category 5.

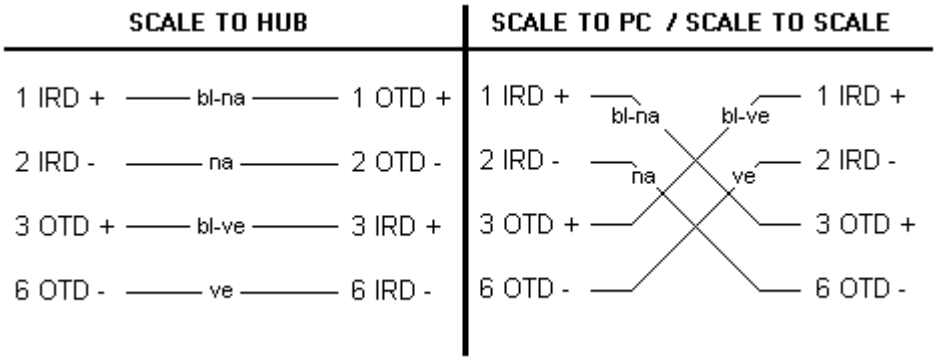
- 22 – 26 AWG, 4 strand gauge twisted pair cable.
- Cable with RJ-45 connectors.
- Maximum length of segment: 100 metres, characteristic impedance from 85 to 111Ω.
- 10 Mbits/s. speed.

When the distance between hubs and scales is big, for instance, if they are on different floors or buildings even, there is a limitation imposed by the maximum length of the UTP cable (100m). If the separating distance is greater than this, UTP cable can be used by siting repeaters every 100m.

Only four (4) of the eight (8) wires in the UTP cable are used for LAND data transfer (two for transmission and two for reception), thus leaving four for other uses (telephony, security systems, video transmission, etc.).

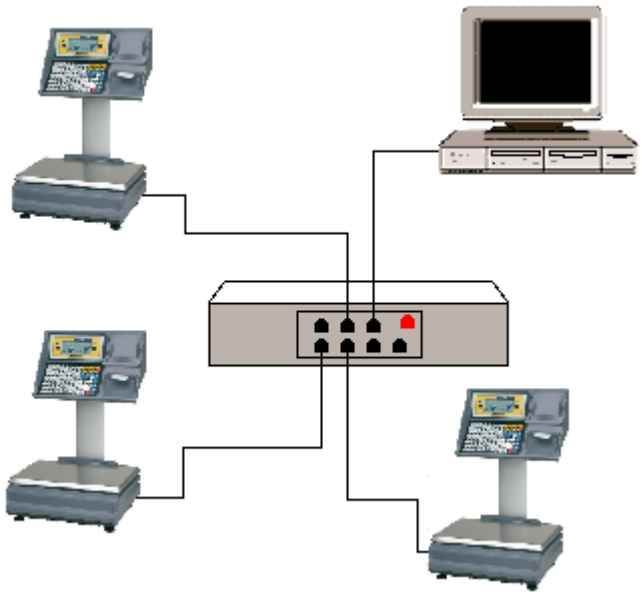
The connector used is similar to that normally used for telephones but having 8 pins instead. It is called an RJ-45 connector. The pins used for data are 1 – 2 for one pair of wires and 3 – 6 for the other. The specification regulating the connection of wires in Ethernet devices is EIA/TIA T568A and T568B.

RJ45 wiring diagram



Star topology: Network configuration used with 10-Base T cabling and a HUB, this cable type is commonly called UTP or twisted pair, where each element is connected to the Hub.

Star topology:

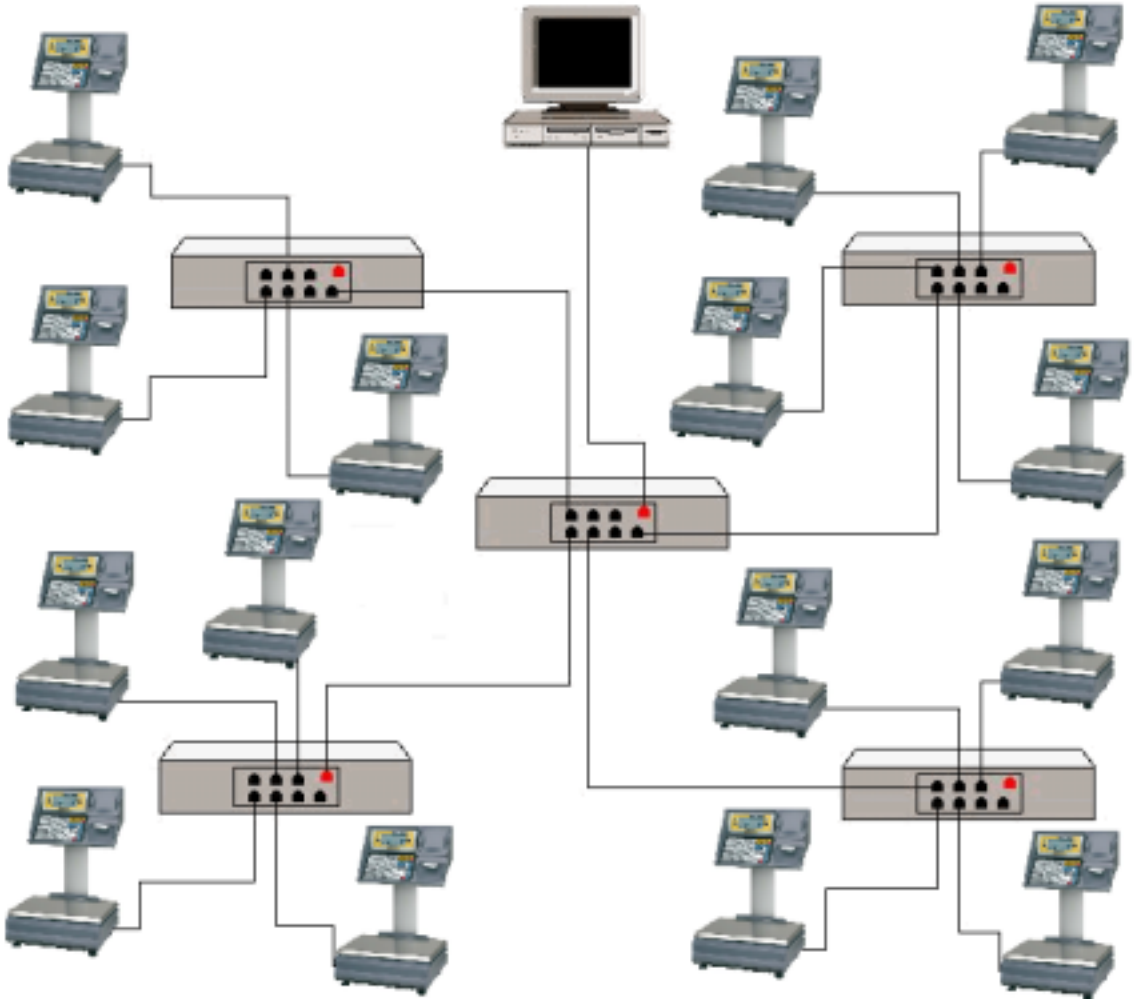


Star topology consists of a central node out of which all connections to the other nodes radiate. All of the information circulating around the network goes through the central node, usually comprised of a hub.

The main advantage here is that it allows all nodes to communicate with each other in a practical manner and that any cable failure or short circuit does not affect the entire network.

In order to increase the number of stations or nodes of the star network, it is not necessary to even partially interrupt the network's activity, the operation can be carried out almost instantly.

Extended star topology



Extended star topology is the same as star topology, the difference being that each node that is connected to the central node is also the centre of another star. As a rule, the central nodes is occupied by a hub or switch and the secondary nodes by hubs.

The advantage of this is that the cable run is shorter and limits how many devices that can be networked with any central node.

RS485 NETWORK

The topology in networks using RS-485 is typically a bus, in which case, there is no central node as is the case in Star topology. All of the nodes (scales) comprising the network are connected to each other in a linear fashion, one after the other.

Bus cabling has fewer logistical problems, given that the cables do not mount up around the central node, as is the case with the Star layout.

The Bus Network must also include devices called terminators or end-of-line caps at both ends of the bus, their purpose being to avoid any possible signal rebound, introducing the characteristic impedance of the line (resistances of 120 Ohms ¼ W.)

Adding new positions to a Bus network means interrupting the network's activity –by section at least. However, this is a fast and simple procedure.

There are minimum regulations that must be followed for laying the communication cable:

1) Do not site cable runs on the outside of buildings.

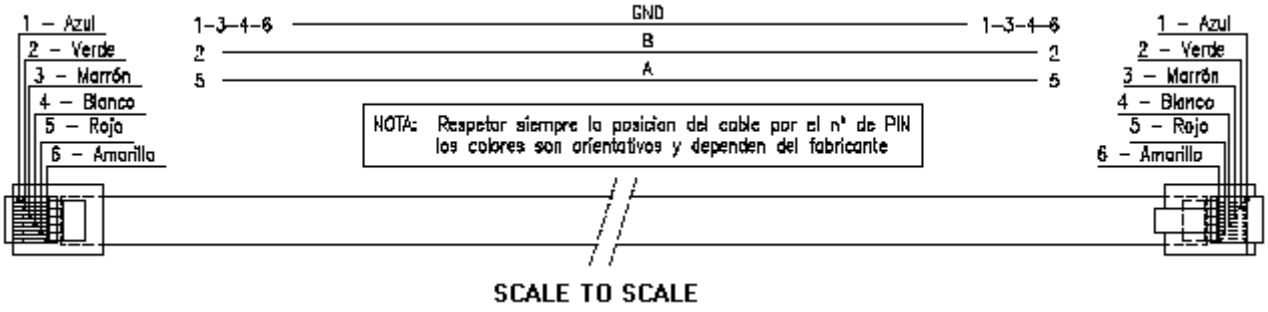
2) Do not lay cable in a conduit used by any other kind of signal carrying cable (telephony, PA system…)

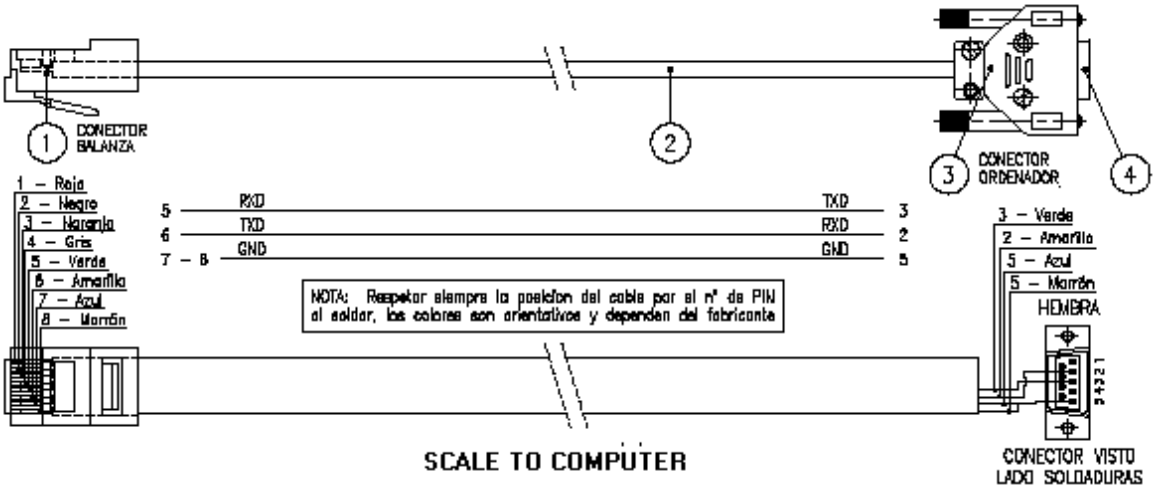
3) To avoid electrical/parasitic interferences, minimum distances must be maintained between the communications cable and electrical power cables, the table below lists said distances:

Less than 2KVA	64 mm
From 2 to 5KVA	152 mm
More than 5KVA	305mm

4) When laying communications cables, these must be protected either using channelling or conduits.

RS-485 Wiring diagram





NOTE: Always respect the position of the cable by the PIM Nº; the colours are orientative and depend on the manufacturer.

Blue Green Brown White Red Yellow

The connection between the computer and scales will be via RS-232, and RS-485 between scales. If the distance between the computer and scales is very big (greater than 12m), it is advisable to use an Ethernet network.

RS-485 bus topology:

Even though the computer is physically connected to the scales via RS-232, the communications protocol used between the computer and the scales will be TCP/IP (SLIP), thus the physical medium does not involve any major change in the protocol.

Ethernet/RS-485 topology:

The BUS network topology is maintained, the scales connected to the computer must be fitted with Ethernet capability, the other scales do not need this given that they are connected to each other via RS-485. The computer has access to the network of scales via ETHERNET in the following way:

2) Xgatdll.Dll library

The XgatDll libraries are a series of subroutines enabling the Gran Máxima scales to be immediately connected to a computer running under the Win32 platform.

- Implemented using Microsoft Visual C++™ 6.0, thereby guaranteeing their portability under Microsoft operating systems.
- Data transfer is done using buffers and ASCII characters compatible with the ANSI standard of Windows.
- By not displaying any pop-up message, it does not impose any specific interface.
- All communications parameters are easily configurable by means of dialogue boxes or function calls.

The aim of the XGat library is to provide a series of tools similar to those provided by the Scale-Mod communication modules. A series of functions enabling the user to transfer information between a PC and the network of scales in a clear and simple fashion.

Data is grouped in the scales in file that are accessible from the read-write functions of the XgatDll library.

The XGat library makes this job easier meaning there is no programming of the ETHERNET port and the information exchange protocols needed, this is also the case for the data received.

The scales comprising a network are organised in section or departments; each section must have a "Master" scales that controls communications, any string of scales can be the "Master". The other scales in the same section will be slaves.

In a system of scales, the sections or departments can be numbered from 0 to 99.

The XgatDll.dll library is supplied in a Win32 version and works perfectly with all the most commonly found programme languages on the market.

3) Command structure of the Maxima scales

Communications between the PC and Scales are via ETHERNET. Communications will take place through the destination IP address of the scales and port 2003. The string contained in the UDP packet, PC/Scales communications string, is the following:

[Dest**] [**Origin**] [**command**] [**file**] [**Register**] [**Parameter**][**Data**]**

- The following fields in bold can be found in many of the commands that will be dealt with: **Destination** and **Destination Nº.** These two fields are compiled in the previous string as follows:

  If the **Destination** fields is Section (S), then **[Dest]** is calculated as 0x80 + **Nº Destination**. If the **Destination** field is Terminal (**T**) **[Dest]** is directly the **Destination** field nº.

- **[Origin]** is the PC, in this case 0.
- **[command]** indicates the operation to be carried out. The following table lists all of the commands corresponding to this field:

COMMAND	LETTER
READ	K
WRITE	L

- **[file]** is the file number that identifies the data of the scales to read from or write to.
- **[Register]** is the option enabling more than one field to be chosen (interval of data to read or write).
- **[Parameter]** is used in some commands for special options.
- **[Data]** is variable depending on the function to be carried out and ends with the control characters **0x0Dh** and **0x0Ah**. (CR and LF, carriage return and line feed). In read commands, the string sent by the PC to request data does not contain the data field, the response for said data, however, from the scales does. In the write commands, the series

sent by the PC to the scales contains the data field, comprised of the data to send to the scales , while the response from the scales to the PC does not have this field.

Some of the fields of the different structures are numerical in nature and can have negative values. These fields include the sign bit and is represented using the ASCII character '-'.

The structures of the commands dealt with below belong to the PC/Scales read and write functions. The messages to be sent are contained in the Data field.

The calls to said commands are parameterised using the fields: **Dest**, **Origin**, **Register** and **Data**. In the case of read commands, the commands end in **r** (read), (**car**, for instance). In the case of write commands, these end in **w** (**write**), (**caw**, for example). There are other kinds of special commands, which do not follow this series format, **clrgt**, for example.

The data sent within the UDP packet complies with the protocol structure between scales. These data are shown in hexadecimal. As an example, the read command for headings is shown, where line 1 of the heading of section 0 is read.

The two bytes whose value is **80**. are the composition of the **Destination** plus **Destination Nº**. **00** indicates the origin, in this case, the PC. **50** indicates the command, in this case, it is read. **00** is the file, in this case, read headings. **01** is the top part of the register and **00** the bottom part. **00** is the segment.

Example:

String sent from the PC to the scales……

80	00	50	00	01	00	00

The response string from the scales to the order sent by the PC, is:

00 01 70 00 01 00 00 **20 20 43 41 52 4e 49 43 41 53 20 4d 55 a5 45 5a 20 53 2e 41 2e 20 20 20**

Where **00** corresponds to the destination where the response is sent to, the PC in this case.

Where **01** corresponds to the source sending the command, the scales in this case. In this case, because reading of the headings from section is asked for, it is the Master that responds. In the event of requesting the reading from a terminal, then the terminal in question will respond.
Where **70** corresponds to the read command from the scales.

Where **00** corresponds to the file.

Where **01** and **00** are the top and bottom parts of the register, in this case, it indicates line 1 of the heading.

Where **00** corresponds to the segment.
The shown in bold **(00 20 20 43 41 52 4e 49 43 41 53 20 4d 55 a5 45 5a 20 53 2e 41 2e 20 20 20)**, correspond to the text associated to line 1 of the heading. The maximum length is 24 ASCII characters (for the receipt heading file).

All of the information contained in the scales is organised using files. Thus, there is a file for products, one for sales, another for keyboard configuration, etc.

Type of access to these files –regardless of whether it be read or write, terminal or section- will depend on the needs of each user.

Thus, for example, it makes sense to write products at section level, given that all of the scales in the same section share the same data. It is therefore not necessary to update each scales in the section. It might, however, be more interesting to programme the keyboard of each scales individually and thus personalise each string of scales.

Whenever a call is made to a library function to access a file, the Terminal 'T' (character 84) or section 'S' (character 83) access mode must be specified.

4) File access functions and their parameters and ranges

The termination "r" or "w" of programmes means THAT the programme READS THE DATA OF THE SCALES (r) or WRITES DATA TO THE SCALES (w).

The names of the functions called in executable format or drivers, will be denominated with the prefix **WM** before the function, e.g.: PLU read function:
DLL function: **PLR**
Function as executable driver: **WMPLR**

```
car/w   : HEADING.
car     [dest] [DestNº] [inireg] [fireg] [buffer]
         (S)     (0)      (0)      (3)
         S/T     0-99     0-5   0-5

plr/w  : PLU FILE.
plw     [dest] [DestNº] [inireg] [fireg] [buffer]
          (S)     (0)      (0)      (400)
          S/T     0-99     0-9999   0-9999

PLIR/W   :READ AND WRITE FILE OF INGREDIENTS
Plir/w    [dest] [DestNº] [inireg] [fireg] [buffer]
          (S)     (0)      (0)      (400)
          S/T     0-99     0-1000   0-1000

chor    : TIME CONTROL.
```

```
chor     [dest] [DestNº] [inireg] [fireg] [buffer]
          (S)     (0)      (0)      (31)
          S/T     0-99     0-31   0-31
```

cdir : DAILY CONTROL.

```
cdir     [dest] [DestNº] [inireg] [fireg] [buffer]
 (S)     (0)      (0)      (31)
 S/T     0-99     0-31   0-31
```

acvnr : VENDORS RUNNING TOTALS.

```
acvnr     [dest] [DestNº] [inireg] [fireg] [buffer]
(S)     (0)      (0)      (11)
S/T     0-99     0-23   0-23
```

acplr : PLU RUNNING TOTALS.

```
acplr     [dest] [DestNº] [inireg] [fireg] [buffer]
         (S)     (0)      (0)      (400)
         S/T     0-99     0-1000   0-1000
```

relr/w : CLOCK SETTINGS.

```
relr     [dest] [DestNº] [buffer]
         (S)      (0)
         S/T      0-99
```

pldr/w : ASSIGNATION OF DIRECT KEYS TO PLUs

```
pldw
pldr     [dest] [DestNº] [inireg] [fireg] [buffer]
         (S)     (0)      (0)      (23)
         S/T     0-99     0-64   0-64
```

famr/w : FAMILY NAMES.

```
famw
famr     [dest] [DestNº] [inireg] [fireg] [buffer]
         (S)     (0)      (0)      (19)
         S/T     0-99     0-99   0-99
```

vnr/w : VENDOR NAMES.

```
vnw
vnr     [dest] [DestNº] [inireg] [fireg] [buffer]
         (S)     (0)      (0)      (23)
         S/T     0-99     0-29   0-29
```

ivar/w : VAT GROUPS.

```
ivaw
ivar     [dest] [DestNº] [inireg] [fireg] [buffer]
       (S)     (0)      (0)      (3)
       S/T     0-99     0-3   0-3
```

cbr/w : BARCODE STRUCTURES.

```
cbw
car     [dest] [DestNº] [inireg] [fireg] [buffer]
         (S)     (0)      (0)      (1)
```

```
              S/T     0-99     0-16   0-16
```

Lotr : BATCH TEXT LINES.

```
      lotr     [dest] [DestNº] [inireg] [fireg] [buffer]
             (S)     (0)      (0)      (0)
             S/T      0-99       0       9
```

cmdr : ORDER FILE.

```
      cmdr     [dest] [DestNº] [inireg] [fireg] [buffer]
               (S)     (0)      (0)      (1)
               S/T     0-99     0-1   0-24
```

mermr : LOSSES OR RETURNS FILE.

```
      mermr     [dest] [DestNº] [inireg] [fireg] [buffer]
               (S)     (0)      (0)      (1)
               S/T     0-99     0-1   0-24
```

pass : TOGGLES ON/OFF THE PASSWORD OF THE MACHINES.

```
      pass     [dest] [DestNº] [buffer]
      (S)     (0)     (0)
      S/T      0-99       0       -999999
```

bloq : BLOCK ALL VENDORS ON THE MACHINES.

```
      bloq    [dest] [DestNº]
               (S)     (0)
               S/T      0-99
```

clrgt : DELETE ALL MACHINE TOTALS AND UNBLOCK (ADVISABLE TO HAVE BLOCKED THEM BEFOREHAND).

```
      clrgt   [dest] [DestNº] [mode]
               (S)     (0)     (1)
               S/T      0-99       0       -3
```

clrcmd : DELETE ORDER INDEX.

```
      clrcmd [dest] [DestNº]
               (S)     (0)
               S/T      0-99
```

clrmerm : DELETE INDEX OF LOSSES OR RETURNS.

```
      clrmerm [dest] [DestNº]
               (S)     (0)
               S/T      0-99
```

TOTIK : READS THE TICKETS FROM SCALES.

```
      TOTIK   [dest] [NºDest] [inireg] [fireg] [buffer] [mode]
               (S)     (0)      (0)      (1)
               S     0-99    0-20000   0-20000
```

5) Opening and closing communications port

Before being able to access any library function, the communications port must be open. Two library functions -detailed below- can be used to manage the opening of the communications port:

- Declaration in MS Visual C++:

```
XGATDLL_API opensocket(char *ipremota, int portlocal, int portdesti,int
ingredients=12);
//Function for opening the communications port.

XGATDLL_API closesocket( );
//Function for closing the communications port.
```

- Declaration in MS Visual Basic:

```
Private Declare Function opensocket Lib “xGatdll.DLL”(ByVal ipremota As String,
ByVal portlocal As Long, ByVal portdesti As Long, ByVal ingredients As Long) As
Long
The value of ingredients by default must be 12 ( 12x24=288 characters).
‘Function for opening the communications port.

Private Declare Function closesocket Lib “xGatdll.DLL”( )As Long
‘Function for closing the communications port.
```

Notes:

1. ipremota is a string of characters indicating the IP address of the scales that needs to be accessed. E.g.- “192.168.2.5”(unicast) “225.0.0.5”(multicast)

2. portlocal and portdesti are two integers indicating the ports to use for the communication with the source and destination respectively.

3. Given the programming of the microprocessor of the MAXIMA scales, the portlocal and portdesti values must be equal, this is because the scales sends the data to the same port via the one it is programmed to receive on.

4. It is advisable to programme the scales’ port with a value of 2003 and up.

5. Each scale has a number of characters for ingredients allowed, this number divided by 24 is reflected in this parameter (ingredients).

CONFIGURATION OF TIMEOUT, RETRIES, ETC.

There is also another function that can be used to set the number of retries in the event of errors or wait time (TIMEOUTS) between retries.

XGATDLL_API settimeouts(int timeout, int nmaxintents,int numeromaq,int traduccio);

The parameters of this function are:

Timeout is the amount of time to wait in the event of an error (value in milliseconds); the value is accepted as modified if greater than –1. The default setting is 6000 (6 seconds).

Nmaxintents is the maximum number of retries; this value is accepted as modified if greater than –1. The default setting is 3 retries.

Numeromaq is the number of machines in the network; this value is accepted as modified if greater than –1. The default setting is 12, this parameter is included to ensure backward compatibility with old DLL versions as is not used for the MAXIMUM (use the –1 value).

Traduccio is the ASCII table type used by the scales, select value 2 (ISO Windows table).

CONFIGURATION OF THE NETWORKED SCALES TABLES

There are two ways of accessing the scales:

1) By means of a multicast IP address that identifies the scales as a network group.
2) By means of a table that specifies each set of scales with its own unicast IP address; detailing the terminal's number, department or section number and whether it is a master or slave.

The network of scales can be identified by means of a multicast IP address (address range between 224.0.0.0 and 239.255.255.255), this means that one single IP address can identify a complete group of scales.

If, on the other hand, the aim is to control each individual set of scales using their own IP address, the unicast IP address system will have to be used to identify each set of scales, in accord with the IP address of the actual network of scales. These can be classified as per the following table:

Class	IP address from	To IP address	Sub-network mask
A	0.0.0.0	127.255.255.255	255.0.0.0
B	128.0.0.0	191.255.255.255	255.255.0.0
C	192.0.0.0	223.255.255.255	255.255.255.0

In the event of using the individual IP address per set of scales identification method, the following function integrated in the DLL can be used in order to design a table that

identifies and links the IP address of each set of scales with its terminal number within the network, and the section or department it belongs to.

Definition in MS Visual C++:

XGATDLL_API AddIPRelTableEntry(int nTerm, int nSecc, int master, char* IP);

This function enables the programmer to enter a link between the terminal and section numbers, master/slave and IP addresses.

This link will be what the programme searches for in the table when sending to that machine. The Master integer can be either 1 or 0, thus indicating whether the scales it refers to is a Master or not.

In the event of wishing to have an IP address as the default route for any set of scales (table not used), the default parameters to send to the function will be as below:

nTerm = 100 nSecc = 100 master = 2 IP address required as entry point to the system.

Configuration of the communications drivers

The communications drivers or executable files, are .EXE files which invoke the functions contained within the DLL (Dynamic Link Library), as is the case with DLL functions, their purpose is to Read (R) or Write (W) data contained in the files of the scales.

All drivers start with the letters WM (Windows Máxima), followed by the name of the file they must access, and end in the letters (R) or (W) depending on whether they are going to read or write respectively.

These drivers interact with the external applications via the use of plain or ASCII text files, using a determined structure detailed in this manual.

The drivers are configured with a file called PARGAT.INI, which contains the parameters to be configured, such as the IP address to access, port or socket number, number of retries, etc.

The file's format is as follows:

IP_DESTI=225.0.0.6

PORT_DESTI=2003
PORT_LOCAL=2003
REINTENTS=3
TIMEOUT=4
DISPLAY=1
DEBUG=0
INGREDIENTS=10

[table]
NUM_ENTRIES=2
sec1=0 term1=0 master1=1 IpAsig1=10.1.0.5
sec2=1 term2=1 master2=1 ipASig2=10.1.0.4

The IP_DESTI field having a value of 225.0.0.6, determines the IP address (multicast, in this instance), to be accessed.

The PORT_DESTI field with the value 2003, determines the input port of the scales.
The PORT_LOCAL field with the value 2003, determines the E/S port of the computer.

The REINTENTS field with the value 3, configures the number of retries.

The TIMEOUT field with the value 4, configures the time in seconds for the timeout.

The DISPLAY field with the value 1, indicates which data to be written or read must be shown on the computer screen. If this is not desired, set the value to 0.

The DEBUG field with the value 0, enables LOG or debug mode (it generates a file called Modulcomm.log that stores all of the communications packets); this field should not be enabled unless needed.

The INGREDIENTS field with the value 10, indicates how many lines of ingredients (24 characters each) are to be used (exclusively for use with driver WMPLIW or WMPLIR).

The identifier of the address table [table] or [TABLE] and the following parameters, are optional and only used when wishing to create a table of scales with its respective IP address and configuration.

The following line is used to specify the number of lines that will follow (these refer to the different entries through which access can be granted to the scales' system), in this instance 2 (NUM_ENTRIES=2).

The different assignations are listed below. Each line has a set of identifiers with an associated number. The identifiers on the first line will be:

SEC1 TERM1 MASTER1 IPASIG1

Those on the second line:
SEC2 TERM2 MASTER2 IPASIG2

Those on the third line:
SEC3 TERM3 MASTER3 IPASIG3

And continues so to the end of the number of lines stipulated in NUM_ENTRIES.

The section will be indicated in the SECX field; the input terminal in TERMX; in MASTER I/O depending on whether it is a Master or not, and the IP address in IPASIG that will be used as the entry for said terminal.

Whenever a driver causes an error, this will be recorded in a file called ORDENES.ERR.

6) DESCRIPTION OF THE COMMANDS

6.1) HEADINGS (car, caw)

Car File: 0 Command: Read
Caw File: 0 Command: Write

```
car/w   : HEADING.
car     [dest] [DestNº] [inireg] [fireg] [buffer]
          (S)     (0)      (0)      (3)
          S/T     0-9     0-5   0-5
```

Declaration in MS Visual Basic:

```
Private Declare Function car Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As Long,
       ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function caw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As Long,
       ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Proceeds to read the receipt headings, referring to the text that will be printed as the heading for each receipt. The scales have 6 lines for receipt headings (0-5), 4 on the top part of the receipt and 2 on the bottom of it.
Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Line nº.	2	5	0-5

SPACE	1	7	Separation space
Text	24	8	Alphanumerical characters

Example:
S 10 00 ***\B\A\L\A\N\C\E\S ****

We have programmed a text on line 1 (register 0 corresponds to 1) of the receipt heading of section or department 10.

6.2) ASSIGNATION OF TEXT TO FAMILIES (famr, famw)

Famr File: 2 Command: Read
Famw File: 2 Command: Write

Text of the family names under which products are grouped. The scales have 100 families (0-99)

Declaration in MS Visual Basic:

```
Private Declare Function famr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function famw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Destination nº.	2	5	0-99
SPACE	1	7	Separation space
Family name	24	8	Alphanumerical characters

Example:

S 00 02 ****** VEGETABLES ********

We have programmed a text in family 2 of section or department 00.

6.3) ASSIGNATION OF DIRECT PLU KEYS: (pldr, pldw)

Pldr File: 4 Command: Read
Pldw File: 4 Command: Write

Setting up of the scales' keyboard. Each key can be individually assigned to a product or vendor. The field type indicates whether the key refers to a PLU (0) or a vendor (1).

Declaration in MS Visual Basic:

```
Private Declare Function pldr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function pldw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Key nº.	4	5	0-according to scales model
SPACE	1	9	Separation space
Assigned PLU nº.	8	10	0-according to scales model
SPACE	1	18	Separation space
Type	1	19	0=PLU 1=Vn

Example:
S 09 0000 00000140 0

We have assigned PLU number 140 to the direct PLU or key 1 (register 0 corresponds to 1) of section or department 9.

6.4) VENDOR'S RUNNING TOTAL (acvnr)

Acvnr File: 7 Command: Read

Declaration in MS Visual Basic:

```
Private Declare Function acvnr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Enables access to the total of all sales made by a vendor. The number of vendors depends on the scales model.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Vendor nº.	2	5	0-according to scales model
SPACE	1	7	Separation space
Credits	10	8	0-according to scales model
SPACE	1	18	Separation space
Credit	12	19	0-according to scales model
SPACE	1	31	Separation space
Card	12	32	0-according to scales model
SPACE	1	44	Separation space
Cheque	12	45	0-according to scales model
SPACE	1	57	Separation space
Voucher	12	58	Explanation needed
SPACE	1	70	Separation space
Total	12	71	Explanation needed
SPACE	1	83	Separation space
Clients	6	84	Amount sold on cards 0-999999
SPACE	1	84	Separation space
Positive cancellations	10	90	Positive cancellations 0-9999999999
SPACE	1	100	Separation space
Negative cancellations	10	101	Negative cancellations 0-9999999999
SPACE	1	111	Separation space
Grammes	10	112	Grammes 0-9999999999
SPACE	1	117	Separation space
RESERVED	8	118	RESERVED

Example:

S 00 00 0000000000 000000000000 000000000900 000000000800 000000000700 000000002400 000400 0000000000 0000000000 00000 0000000000

The sales data indicated is shown in vendor 1 (0 corresponds to 1) of section or department 0.

6.5) PLU RUNNING TOTALS (acplr)

Acplr File: 8 Command: Read

Read only file, enables access to the total of all sales of a product or range of specific products. The number of products depends on the model of scales.

Declaration in MS Visual Basic:

```
Private Declare Function acplr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
PLU nº.	8	5	0-according to scales model
SPACE	1	13	Separation space
Grammes	10	14	0-99999999
SPACE	1	24	Separation space
Total amount	10	25	-99999999 to 99999999
SPACE	1	35	Separation space
Operations	6	36	0-9999
SPACE	1	42	Separation space
Packets	6	43	(0-999999)
SPACE	1	49	Separation space
Stock (Kg/Pk)	8	50	-9999999 to 99999999 Kg -9999999 to 99999999 Packets

Example:

S 00 00000001 0000000000 0000999999 000001 000001 00000060

The sales data indicated is shown in PLU 1 of section or department 0.

6.6) DAILY CONTROL (cdir)

Cdir File: 9 Command: Read

Read only file, gives daily partial sales totals. *Note: The buffer is revolving (32 registers), and when full, registers drop out starting with register 0. A new register is generated whenever a Grand Total is carried out.*

Declaration in MS Visual Basic:

```
Private Declare Function cdir Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Register nº.	2	5	0-according to scales model
SPACE	1	7	Separation space
Day	2	8	1-31
SPACE	1	10	Separation space
Month	2	11	1-12
SPACE	1	13	Separation space
Year	4	14	0-9999
SPACE	1	18	Separation space
Amount	12	19	0- 999999999999
SPACE	1	31	Separation space
CLR vendors	1	32	Flag indicating a vendors grand total has been carried out 0 = NO 1=YES
SPACE	1	33	Separation space
CLR PLUs	1	34	Flag indicating a vendors grand total has been carried out 0 = NO 1=YES

Example:
S 00 29 10 10 2000 000000050279 1 1

This example shows the reading of the daily control after having carried out a Grand Total (vendors and products).

6.7) HOURLY CONTROL (chor)

Chor File: 10 Command: Read

Read only file, revolving 32-register buffer providing partial sales made every hour. *Note: Every new hour, the registers move down the buffer. Register 0 being eliminated.*

Declaration in MS Visual Basic:

```
Private Declare Function chor Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Register nº.	2	5	0-according to scales model
SPACE	1	7	Separation space
Hour	2	8	1-23
SPACE	1	10	Separation space
Day	2	11	1-31
SPACE	1	13	Separation space
Month	2	14	1-12
SPACE	1	16	Separation space
Year	4	17	0-9999
SPACE	1	18	Separation space
Amount	12	19	0- 999999999999

Example:
S 00 30 10 11 10 2000 000000004869

6.8) CLOCK DATA (relr, relw)

Relr File: 20 Command: Read
Relw File: 20 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function relr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long

Private Declare Function relw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

File containing the actual time of the scales.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Seconds	2	5	0-59
SPACE	1	7	Separation space
Minutes	2	8	0-59
SPACE	1	10	Separation space
Hours	2	11	0-23
SPACE	1	13	Separation space
Day	2	14	1-31
SPACE	1	16	Separation space
Month	2	17	1-12
SPACE	1	19	Separation space
Year	4	20	0000-9999

Example:

S 00 00 12 10 04 06 2003

6.9) PLUs (plr, plw)

Plr File: 1 Command: Read
Plw File: 1 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function plr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function plw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

File containing information about the products. The number of products depends on the model of scales.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
PLU nº.	8	5	0-according to scales model
SPACE	1	13	Separation space
Name	24	14	0-according to scales model
SPACE	1	38	Separation space
Price	6	39	0 -999999
SPACE	1	45	Separation space
Offer mode	1	46	Offer types
SPACE	1	47	Separation space
Offer 1	6	48	0 -999999
SPACE	1	54	Separation space
Offer 2	6	55	0 –999999
SPACE	1	61	Separation space
Tare	6	62	0- 999999
SPACE	1	68	Separation space
Code	6	69	0 –999999
SPACE	1	75	Separation space
Section	2	76	0-99
SPACE	1	78	Separation space
Family	2	79	0-99
SPACE	1	81	Separation space
VAT	1	82	0-3
SPACE	1	83	Separation space

Mode	1	84	0=Indistinct 1=Weight 2=E+ 3=E-
SPACE	1	85	Separation space
Units per packet	2	86	0-99 (reserved)
SPACE	1	88	Separation space
Blocking	1	89	Blocking modification of price on scales 0- Not blocked 1- Blocked
SPACE	1	90	Separation space
Traceability type	1	91	0 – none, 1-bovine, 2-fish
SPACE	1	92	Separation space
Traceability of batch	2	93	0-31
SPACE	1	95	Separation space
Label format	2	96	0-99
SPACE	1	98	Separation space
Expiry date	6	99	DD/MM/YY
SPACE	1	105	Separation space
Consume by date	6	106	DD/MM/YY (reserved)
SPACE	1	112	Separation space
Expiry time	2	113	0-24 (reserved)
SPACE	1	115	Separation space
Modo_Ean	2	116	Mode Ean, 0-20
SPACE	1	118	Separation space
Ean	14	118	Ean structure:
SPACE	1	133	Separation space
Losses	2	134	% reduction frozen, drained weight 0 -99
SPACE	1	136	Separation space
Modified	1	137	1= YES 0=NO

*Reserved : future features

Example:

S 07 0009999 RIPE CHOPPED TOMATO 1 000100 000075 000050 000010 123456 09 39 1 1 02 1 0 00 25 000030 6 00 00841254123567 00 1

The data corresponding to said product are shown in PLU 9999 of section or department 7.

6.10) PLU INGREDIENTS (plir, pliw)

Plir File: Instruction: Read
Pliw File: Instruction: Write

Declaration in MS Visual Basic:

```
Private Declare Function plir Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function pliw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

File containing the ingredients of the products. The lines of ingredients is comprised of a string of characters 1024 bytes ASCII in length (this value may vary according to model of scales).

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
PLU nº.	8	12	0-according to scales model
SPACE	1	13	Separation space
Text	1024 (according to scales model	14	Alphanumerical characters

Example:

S 00 0009999 Flour, Apples, Sugar ...

The texts of the ingredients of said product are shown in PLU 9999 of section or department 0.

6.11) DELETE PLU (delplu)

Delplu File: 1 Command: Read

Declaration in MS Visual Basic:

```
Private Declare Function delplu Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long) As Long
```

This command deletes a PLU from the scales.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
PLU nº.	8	5	0-according to scales

Example:
S 00 0009999

6.12) VAT GROUPS (ivar, ivaw)

Ivar File: 33 Command: Read
Ivaw File: 33 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function ivar Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function ivaw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

This register contains the value of the different VAT groups. There are 4 VAT groups on the scales. In the example given, the last bytes the value of which is **42**, are the bytes indicating the decimal part of the VAT. The two previous bytes whose value is **16**, are the integer component of the VAT.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space

VAT number	2	5	0- 3
SPACE	1	7	Separation space
Percentage	4	8	Numerical characters

Example:

S 00 01 1642

6.13) BARCODE STRUCTURE (cbr, cbw)

Cbr File: 28 Command: Read
Cbw File: 28 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function cbr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function cbw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

This file contains the barcode structure. The scales have ten different barcode structures.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Barcode nº.	2	5	0- 9
SPACE	1	7	Separation space
Structure	24	8	Alphanumerical characters

Configuration of the code specified in the "**Structure**" field can be done as per the options listed below:

Letter	Meaning	Max. length	Observations
0…9	Fixed numbers	12	
A	Terminal number	2	Only on networked
B	Section	1	Only on networked
Alphanumerical characters	Vendor	2	Vendor number
D	Vendor receipt nº.	4	

E	Correlative nº.	6	Correlative number of receipt
F	Operation type	1	0- Programmed as indistinct 1- Programmed as weighing 2- Not operative 3- Programmed as external positive 4- Programmed as external negative
G	Weight	6	With overrange detection
H	Price	6	With overrange detection
I	Amount (on label)	6	With overrange detection
J	Receipt total	8	With overrange detection
K	PLU number	6	
L	Associated code	6-12	
M	Identification of operations	1	0,1,2,3 (0,2 positive amount) (1,3 negative amount)
O	Intermediate Checksum	1	Only on 7th and 8th position
P	Weight / 10	6	With overrange detection
Q	Total weight on types 2 and 3 receipts	8	With overrange detection
R	Customer number	6	Only with customer function enabled
S	Family number	2	
T	VAT number	1	
U	2nd currency on label	6	With overrange detection
V	2nd currency on label	6	With overrange detection
X	Packets * 1000	6	With overrange detection

Example:
S 00 00 0028CCCCJJJJJJ

The data indicated is shown in the barcode structure 0 of section or department 0.

6.14) ADVERTISING: (fimr, fimw)

Fimr File: 34 Command: Read
Fimw File: 34 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function fimr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function fimw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

This file refers to the description of the text that appears in the advertising display. There are 4 lines for advertising text.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Nº of advertising text line	2	5	0-3
SPACE	1	7	Separation space
Text	24	8	Alphanumerical characters

Example:

S 00 02 Polígono Cova solera s/n

The programmed advertising text is presented and displayed on line 3 (2 corresponds to 3) of section or department 0.

6.15) VENDORS NAMES (vnr, vnw)

Vnr File: 35 Command: Read

Vnw File: 35 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function vnr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function vnw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

File containing the description of the vendors.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Vendor nº.	2	5	0- 29

SPACE	1	7	Separation space
Text	24	8	Alphanumerical characters

Example:

S 00 00 JOHN has attended you.

6.16) BATCH TEXT (Lotr, Lotw)

Lotr File: 51 Command: Read
Lotw File: 51 Command: Write

Register in which the Batch text to appear on the receipt can be read/written.

Declaration in MS Visual Basic:

```
Private Declare Function lotr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Batch nº.	2	5	0- 5
SPACE	1	7	Separation space
Batch	24	8	Alphanumerical characters

Example:

S 00 00 Batch Number A-7/25

6.17) DELETING LIST OF ORDERS (clrcmd)

Clrcmd File: 46 Command: Write

This command deletes the list of orders.

Declaration in MS Visual Basic:

```
Private Declare Function clrcmd Lib "xGatDll.DLL" (ByVal dest As Long,
        ByVal ndest As Long) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Option	1	5	Blocking of entering orders 0 Free 1 Blocking

Example:
S 00 0

6.18) ORDERS (cmdr)

Cmdr File: 46 Command: Read

This file contains the different registers produced as orders.

Declaration in MS Visual Basic:

```
Private Declare Function cmdr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
File nº.:	2	5	0- 99
SPACE	1	7	Separation space
PLU nº.	8	8	0-according to scales
SPACE	1	16	Separation space
Date	12	17	DD/MM/YYYY/HH/mm

SPACE	1	29	Separation space
Vendor	2	30	0-24
SPACE	1	32	Separation space
Quantity	6	33	0-999999
SPACE	1	39	Separation space
Unit type	2	40	00= Packets, 01 =Kg
SPACE	1	42	Separation space
Status	2	43	Current status of order 0 = Not blocked 99 = Blocked

Example:
S 00 01 00002000 071219991322 02 000050 00 00

Note: This function must read each register in turn until finding the last one, which will have all of its fields string at 0. This will indicate there is no further order.

6.19) DELETING LIST OF LOSSES OR RETURNS (clrmerm)

Clrmerm File: 48 Command: Write

This command deletes the list of receipt headings.

Declaration in MS Visual Basic:

```
Private Declare Function clrmerm Lib "xGatDll.DLL" (ByVal dest As Long,
        ByVal ndest As Long) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Option	1	5	Blocking of losses 0 Free 1 Blocking

Example:
S 00 00

6.20) RETURNS OR LOSSES (mermr)

Mermr File: 48 Command: Read

This file contains the different registers produced as returns or losses.

Declaration in MS Visual Basic:

```
Private Declare Function cmdr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
File nº.:	2	5	0- 99
SPACE	1	7	Separation space
PLU nº.	8	8	0-according to scales
SPACE	1	16	Separation space
Date	12	17	DD/MM/YYYY/HH/mm
SPACE	1	29	Separation space
Vendor	2	30	0-24
SPACE	1	32	Separation space
Packets	6	33	0-999999
SPACE	1	39	Separation space
Id. data	2	40	00= Packets, 01 =Kg 02 = Kg returned, 02 = Packets returned
SPACE	1	42	Separation space
Reserved	2	43	RESERVED

Example:
S 00 01 00000019 071219991322 02 000050 00 00

Note: This function must read each register in turn until finding the last one, which will have all of its fields string at 0.

6.21) READ CONFIGURABLE LABEL (labelr)

Labelr File: 47 Command: Read

This command reads the configurable label and saves it in an external file with a BIN extension. The external file is divided and sent to the scales in blocks of 32 bytes. The external binary file has a fixed size of 1024 bytes (its structure is defined in the label configuration document).

Declaration in MS Visual Basic:

```
Private Declare Function Labelr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Label nº.	2	5	0-59
SPACE	1	7	Separation space
Label	32	8	Binary data

Returns a binary file with the label

6.22) WRITING OF CONFIGURABLE LABEL (labelw)

Labelw File: 47 Command: Write

This command sends the configurable label to the scales. This command sends the configurable label via an external file with a BIN extension. 32 byte blocks are sent in each string. The external binary file has a fixed size of 1024 bytes (its structure is defined in the label configuration document).

Declaration in MS Visual Basic:

```
Private Declare Function Labelw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Label nº.	2	5	0- 59
SPACE	1	7	Separation space

Label	32	8	Binary data

Example:
S 00 02 label02.bin

6.23) DELETE CONFIGURABLE LABEL (labeler)

Labeler File: 47 Command: Write

This command deletes a configurable label.

Declaration in MS Visual Basic:

```
Private Declare Function Labeler Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Label nº.	2	5	0- 59

Example:
S 00 02

6.24) READ TOP LOGO (logor)

Logor File: 38 Command: Read

This command reads the top logo of the scales. The extension of the external file is BMP in black and white. 54 byte blocks are read in each string, corresponding to each of the lines of the logo. The logo read of the scales is saved in an external file on a disk.

Declaration in MS Visual Basic:

```
Private Declare Function Logor Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space

Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Logo	54	5	Size of logo above 432 x 96

Returns a BMP (graphic logotype):

6.25) WRITE TOP LOGO (logow)

Logow File: 38 Command: Write

This command sends the top logo to the scales. The logo to be written to the scales comes from an external file saved on a disk. The extension of the external file is BMP in black and white. 54 byte blocks are sent in each string.

Declaration in MS Visual Basic:

```
Private Declare Function Logow Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Logo	54	5	Size of logo above 432 x 96

Example:
S 01 logsup.bmp

6.26) WRITE BOTTOM LOGO (logow)

Formw File: 39 Command: Write

This command sends the bottom logo to the scales. The logo to be written to the scales comes from an external file saved on disk. The extension of the external file is BMP in black and white. 54 byte blocks are sent in each string.

Declaration in MS Visual Basic:

```
Private Declare Function Formw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Logo	54	5	Size of logo above 432 x 512

Example:
S 01 loginf.bmp

6.27) READ BOTTOM LOGO (formr)

Formw File: 39 Command: Write

This command reads the bottom logo of the scales. The logo read of the scales is saved in an external file on disk. The extension of the external file is BMP in black and white. 54 byte blocks are sent in each string.

Declaration in MS Visual Basic:

```
Private Declare Function Formr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Logo	54	5	Size of logo above 432 x 512

Example:
S 01 loginf.bmp

6.28) PLU PRICES (prplr, prplw)

Prplr File: 50 Command: Read
Prplw File: 50 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function Prplr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function Prplw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Register in which product prices can be read/written. This function is of interest if only wishing to update prices without the need to transfer the entire product structure.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
PLU nº.	8	5	0-according to scales model
SPACE	1	13	Separation space
Price	6	14	0 -999999

Example PLU 1:
S 00 12345678 000125

6.29) BATCH FOR BOVINE TRACEABILITY (tlotr, tlotw)

Tlotr File: 51 Command: Read
Tlotw File: 51 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function Tlotr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function Tlotw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

Register containing the structure of the batch for BOVINE traceability. There are 32 traceability lots (from 0 to 31).)

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99

SPACE	1	4	Separation space
Batch nº.	2	5	0- 31
SPACE	1	7	Separation space
DescripciónDescription	16	8	Alphanumerical characters
SPACE	1	24	Separation space
Country of origin	2	25	0-15
SPACE	1	27	Separation space
Country reared in	2	28	0-15
SPACE	1	30	Separation space
Country slaughtered in	2	31	0-15
SPACE	1	33	Separation space
Country butchered in	2	34	0-15
SPACE	1	36	Separation space
Date of birth	6	37	Numerical characters DD/MM/YY
SPACE	1	43	Separation space
Date slaughtered	6	44	Numerical characters DD/MM/YY
SPACE	1	50	Separation space
Category	2	51	0-15
SPACE	1	53	Separation space
Breed	2	54	0-15
SPACE	1	56	Separation space
Slaughterhouse	2	57	0-15
SPACE	1	59	Separation space
Butchering hall	2	60	0-15
SPACE	1	62	Separation space
Default batch	2	63	0-31

Example:
S 00 00 ABCDEFGHIJ123456 01 14 04 03 160199 250401 01 05 02 04 01

6.30) DESCRIPTION OF BREEDS FOR BOVINE TRACEABILITY (racr, racw)

Racr File: 54 Command: Read
Racw File: 54 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function Racr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function Racw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
```

Long, **ByVal** inireg **As** Long, **ByVal** fireg **As** Long, **ByVal** buf **As** String) **As** Long

This register contains the descriptions of the different breeds of animals. There are 16 breed registers (from 0 to 15))

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Line nº.	2	5	0-according to scales model
SPACE	1	7	Separation space
Text	12	8	Alphanumerical characters

Example:
S 00 00 HORSE STD.

6.31) DESCRIPTION OF COUNTRIES FOR BOVINE TRACEABILITY (cor, cow)

Cor File: 52 Command: Read
Cow File: 52 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function Cor Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function Cow Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

This register contains the descriptions of the different countries of origin, rearing, slaughtering or butchering of the animals. There are 16 breed registers (from 0 to 15).

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Line nº.	2	5	0-according to scales

			model
SPACE	1	7	Separation space
Text	14	8	Alphanumerical characters

Example:

S 00 00 SPAIN ES

6.32) DESCRIPTION OF CATEGORIES FOR BOVINE TRACEABILITY (catr, catw)

Catr File: 53 Command: Read
Catw File: 53 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function Catr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function Catw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

This register contains the descriptions of the different categories of animals. There are 16 breed registers (from 0 to 15).

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Line nº.	2	5	0-according to scales model
SPACE	1	7	Separation space
Text	12	8	Alphanumerical characters

Example:
S 00 00 MEDIA TRADIT

6.33) DESCRIPTION OF SLAUGHTERHOUSES FOR BOVINE TRACEABILITY (slar, slaw)

Slar File: 55 Command: Read
Slaw File: 55 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function Slar Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function Slaw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

This register contains the descriptions of the different slaughterhouses in which the animal has been slaughtered. There are 16 breed registers (from 0 to 15).

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Line nº.	2	5	0-according to scales model
SPACE	1	7	Separation space
Text	12	8	Alphanumerical characters

Example:
S 00 00 MERCABARNA 1

6.34) DESCRIPTION OF BUTCHERING HALLS FOR BOVINE TRACEABILITY (despr, despw)

Catr File: 56 Command: Read
Catr File: 56 Command: Write

Declaration in MS Visual Basic:

```
Private Declare Function Despr Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long

Private Declare Function Despw Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String) As Long
```

This register contains the descriptions of the different butchering halls in which the animal has been prepared. There are 16 breed registers (from 0 to 15).

Structure:

Denomination	Number of bytes	Location	Range

Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Line nº.	2	5	0-according to scales model
SPACE	1	7	Separation space
Text	12	8	Alphanumerical characters

Example:
S 00 00 SOUTH-1 SECTOR

6.35) STRUCTURE OF RECEIPTS EXTERNAL FILE (totik)

Declaración en MS VISUAL BASIC:

```
Private Declare Function Totik Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal inireg As Long, ByVal fireg As Long, ByVal buf As String, ByVal borrado
As Long ) As Long
```

The MASTER scale returns only one file, on this file we have all the needed information to construct one ticket, this file is one ASCII file with a fixed and simetric structure (the same for the ticket header and his operations).

In this file we have the ticket and his operations (the header is always placed before his operations and the last field indicates the operations number). There is a byte (NUMBER 9) wich stablish the difference between one ticket header line and one operation line.

The driver gets the tickets by section (the master scale gets all the tickets of the master and his slaves, the scales also works as sub-masters and backup of his master). The memory for tickets and operatioons is circular and the capacity are between 10.000 and 20.000 (depending of the memory RAM of the scales).

The TOTIK function is executed in this way : TOTIK S 0 0 20000 TOTIK.ASC 1

Where S= SECTION
0= SECTION NUMBER
0= first line
20000=last line
TOTIK.ASC= file or buffer for the data
1= parameter

PARAMETROS:

0 = Reads the tickets and don't erase them.
1 = Reads the tickets and erase them
2 = Only erase the tickets
3 = Read only the tickets

Usually we program one read interval from 0 to 20.000 to read all the lines on the memory of the scale in that moment. The function totik will return the ERROR 8 when finish (the function finish the last available line).

Data structure:

Data structure for the ticket header

Denomination	Characters	Comments
Destination	1	
SPACE	1	
Number of destination	2	
SPACE	1	
Line number	6	
SPACE	1	
Vendor	2	
SPACE	1	
Operative Type	1	'9' IF IS A TICKET HEADER LINE
SPACE	1	
Total +	8	
SPACE	1	
Total -	8	
SPACE	1	
RESERVED	10	
SPACE	1	
Payment mode	1	
SPACE	1	
RESERVED	6	
SPACE	1	
RESERVED	6	
SPACE	1	
Correlative number	6	
SPACE	1	
Minutes	2	
SPACE	1	
Hour	2	
SPACE	1	
Day	2	
SPACE	1	
Month	2	
SPACE	1	
Year	2	
SPACE	1	
RESERVED	6	
SPACE	1	
RESERVED	1	
SPACE	1	
RESERVED	6	
SPACE	1	
RESERVED	1	
SPACE	1	
Number of operations	5	

SPACE	1	
Work mode of the ticket	1	
SPACE	1	
Terminal number	2	

Structure for operation line

Denomination	Characters	Comments
Destination	1	
SPACE	1	
Number of destination	2	
SPACE	1	
Number of line	6	
SPACE	1	
Vendor	2	
SPACE	1	
Operative type	1	
SPACE	1	
Weight or packets	8	
SPACE	1	
Price	8	
SPACE	1	
Amount	10	
SPACE	1	
Payment mode	1	
SPACE	1	
PLU	6	
SPACE	1	
PLU code	6	
SPACE	1	
Correlative number	6	
SPACE	1	
Minutes	2	
SPACE	1	
Hour	2	
SPACE	1	
Day	2	
SPACE	1	
Month	2	
SPACE	1	
Year	2	
SPACE	1	
RESERVED	6	
SPACE	1	
Cancelled	1	
SPACE	1	
Tare	6	
SPACE	1	
Tare auto/manual	1	Auto = ’1’, manual = ‘0’
SPACE	1	
RESERVED	5	
SPACE	1	

RESERVED	1	
SPACE	1	
Terminal number	2	

7) SPECIAL COMMANDS

All of the following commands are special because they do not follow the string previously seen up to now.

7.1) UNBLOCKING AND GRAND TOTAL (clrgt)

Clrgt File: 74

Declaration in MS Visual Basic:

```
Private Declare Function Clrgt Lib "xGatDll.DLL" (ByVal dest As Long,
        ByVal ndest As Long, ByVal option As Long) As Long
```

Enables a previously blocked section to be unblocked or a Grand Total to be carried out. The option value determines the type of Grand Total:

0 .- unblock the vendors
1.- reset vendors
2.- reset PLU
3.- reset vendors + PLU
4.- reset tickets
5.- reset vendors + tickets
6.- reset PLU + tickets
7.- reset vendors + PLU + tickets
8.- reset storehouse
9.- reset vendors + storehouse
10.- reset PLU + storehouse
11.- reset PLU + vendors + storehouse
12.-reset tickets + storehouse
13.- reset vendors + tickets + storehouse
14.- reset PLU + tickets + storehouse
15.- reset vendors + PLU + ticket + storehouse

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99
SPACE	1	4	Separation space
Option	1	5	0- 3

Example:
S 00 1

7.2) BLOCKING (bloq)

Bloq File: 72

Declaration in MS Visual Basic:

```
Private Declare Function bloq Lib "xGatDll.DLL" (ByVal dest As Long,
        ByVal ndest As Long) As Long
```

This command blocks all of the vendors of a section. It is advisable to execute blocking prior to carrying out the Grand Total.

Structure:

Denomination	Number of bytes	Location	Range
Destination	1	0	S/T
SPACE	1	1	Separation space
Destination nº.	2	2	0-99

Example:
S 00

7.3) PASSWORD (pass)

Pass File: 79

When the password function is enabled on the scales, no operations can be carried out without having first sent the ACCESS CODE command with the correct password number.

Declaration in MS Visual Basic:

```
Private Declare Function pass Lib "xGatDll.DLL" (ByVal dest As Long, ByVal ndest As
        Long, ByVal buf As String) As Long
```

8) Prototypes of the functions in “C” programming language.

This section shows the call convention of the functions for programmers in C.

```
#define __DSExportStd __declspec(dllexport) __stdcall
#define XGATDLL_API int __DSExportStd
```

```
XGATDLL_API opensocket(char *ipremota, int portlocal, int portdesti);
XGATDLL_API closesocket( );

XGATDLL_API car(int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API caw(int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API plr (int dest, int ndest, int inireg, int fireg, unsigned char *buf, int
nrevia);
XGATDLL_API plw (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API famr (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API famw (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API pldr (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API pldw (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API acvnr (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API acplr (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API cdir (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API chor (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API relr (int dest, int ndest, unsigned char *buf);
XGATDLL_API relw (int dest, int ndest, unsigned char *buf);
XGATDLL_API cbr (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API cbw (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API ivar (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API ivaw (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API fimr (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API fimw (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API vnr (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API vnw (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API lotr (int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API bloq (int dest, int ndest);
XGATDLL_API clrgt (int dest, int ndest, int nrevia);
XGATDLL_API cmdr(int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API mermr(int dest, int ndest, int inireg, int fireg, unsigned char *buf);
XGATDLL_API clrcmd (int dest, int ndest);
XGATDLL_API clrmerm (int dest, int ndest);
XGATDLL_API totik(int dest, int ndest, int inireg, int fireg, unsigned char *buf, int
delete);
```

9) Return values of the functions

0: The operation has finished properly.
2: Source file does not exist.
1: It was not possible to open socket.
3: TimeOut error. (The scales have not responded).
6: Nack error. (Wrong command sent).
8: Checksum error. (There are communications problems).

10) Annexe for Visual Basic programmers

10.1) Porting strings to a DLL procedure

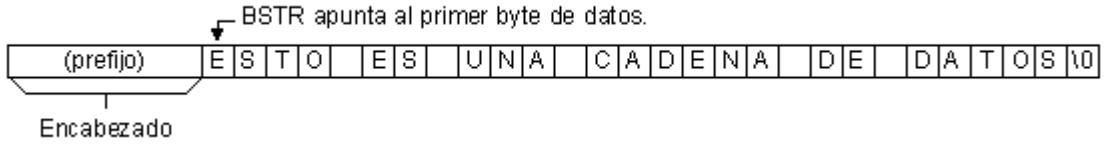
As a rule, **ByVal** must be used to port strings to the API. Visual Basic uses a **String** data type known as BSTR, which is a data type defined by Automation (previously known as OLE). One kind of BSTR data type is formed by a heading containing information about

the length of the string, plus the string itself that may contain null values. This type of data is processed as a flag, hence the DLL procedure is able to modify the string. (A *flag* is a variable containing the memory address of another variable instead of the actual data). The BSTR data are Unicode, which means each character fills two bytes. Generally speaking, the BSTR end with a zero character of two bytes.

Figure 1.2 The BSTR data type (each box represents two bytes).

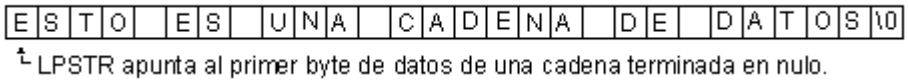
(prefix)BSTR notes the first byte of data.

Heading



Procedures of the majority of DLL files (and of all Windows API procedures) recognise LPSTR types, these being flags of C strings ending in a zero character (also called ASCIIZ strings). LPSTR types have no prefix. The figure below shows an LPSTR type pointing to an ASCIIZ string.

Figure 1.3 LPSTR type.



BSTR notes the first byte of data of a chain ending in zero.

If a DLL procedure is waiting for an LPSTR (a flag in a string ending with a zero character) as an argument, it passes on BSTR type data by default. Given that the BSTR type flags are flags pointing to the first byte of data of the string ending in a zero character, it will look like an LPSTR type to the DLL procedure.

For example:, the **sndPlaySound** function accepts a string assigning a name to a digitalised sound file (.wav) and plays that file.

> *Private Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal lpszSoundName As String, ByVal uFlags As Long) As Long*

Given that the string argument for this procedure is declared through **ByVal**,

Visual Basic issues a BSTR type that points to the first byte of data:

> *Dim SoundFile As String, ReturnLength As Long*
> *SoundFile = Dir("c:\Windows\System\" & "*.wav")*
> *Result = sndPlaySound(SoundFile, 1)*

It generally uses the keyword **ByVal** when issuing string arguments to DLL procedures awaiting LPSTR type strings. If the DLL file is waiting for a flag to an LPSTR type string, it issues the Visual Basic string by default.

If it sends binary data to a DLL procedure, it sends a variable as matrix of the **Byte** data type instead of sending a **String** variable. It is assumed that the strings contain characters and the binary data cannot be properly read in external procedures if sent as **String** variables.

If a string variable is declared without initialising and is then sent as a value to a DLL file, this variable will be sent as NULL, not as an empty (“”) string. To avoid confusion in the code, use the **vbNullString** constant to send a NULL value to an LPSTR argument.

10.2 Sending strings to DLL libraries using Automation

Some DLL’s are specifically written to work with Automation data types, BSTR for instance, using the procedures facilitated by Automation.

Given that Visual Basic uses Automation type data as its own data types, Visual Basic arguments can be sent as reference to any DLL awaiting Automation type data. Therefore, if a DLL procedure is awaiting a Visual Basic string as argument, there is no need to declare the argument with the keyword **ByVal**, unless the procedure specifically needs to procedure the string by values.

Some DLL procedures can return strings to the call procedure. A DLL function cannot return strings unless it has been specifically written to be used with Automation type data. If that is the case, the DLL file will probably provide a library of types detailing the procedures. Refer to documentation appertaining to that DLL file.

To obtain more information about the Automation type data, refer to *OLE 2 Programmer's Reference*, published by Microsoft Press.

10.3 Procedures modifying string arguments

A DLL procedure can modify the data of a string variable it receives as argument. Nevertheless, if the data modified exceeds the length of the original string, the procedure will go beyond the end of the string and probably damage other data.

This problem can be avoided if sufficient length is applied to the string’s argument to ensure that the DLL file’s procedure does not exceed its end. For example: the **GetWindowsDirectory** procedure returns the path of the directory of

Windows in the first argument:

```
Declare Function GetWindowsDirectory Lib "kernel32" Alias
"GetWindowsDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long
```

A secure way of calling up this procedure is to first use the **String** function to define a length of at least 255 characters for the argument returned with null filling characters (binary zero):

Path = String(255, vbNullChar)
ReturnLength = GetWindowsDirectory(Path, Len(Path))
Path = Left(Path, ReturnLength)

Another solution to define the string with a fixed length is as follows:

*Dim Path As String * 255*
ReturnLength = GetWindowsDirectory(Path, Len(Path))

Both procedures give the same result: they create a fixed length string with the longest possible string that can return the procedure.

Note: Windows API DLL processes do not generally expect string buffers whose length is greater than 255 characters. Although this is so with many other libraries, always refer to the procedure documentation.

If the DLL procedure calls up a memory buffer, it may use the appropriate data types or a **Byte** data type matrix.

11) Programming annexe of the communication parameters of the Máxima scales

11.1) Programming of the IP address of the scales

The relevant function must be accessed in order to configure the basic communication parameters of the scales. Access is granted using the password given below when the set of scales is running the display test.

The scales' display will show the IP unicast IP address configuration of the scales (IP address that individually identifies the scales). Use the numerical keypad to enter the corresponding IP address, according to the network that needs to be joined.

And press the ENTER key ↵

The mask selection will then be displayed:

255.255.255.0
IP MASK

And press the ENTER key ↵

The scales' display will show the IP multicast IP address configuration of the scales (IP address that identifies a group of scales). Use the numerical keypad to enter the corresponding IP address. All of the scales must be programmed with the same multicast IP address.

224.0.0.005
IP MULTICAST

And press the ENTER key ↵

Next, and lastly, the port or socket will be displayed, 2000 by default (the default port on the computer is 2003). Modifying this parameter is not recommended unless it is necessary.

2000
SOCKET

And press the ENTER key ↵

11.2) Programming of the communication parameters of the scales

Once programming of the IP multicast, mask of the IP unicast, IP multicast and socket has been completed, access will then be granted to the function for programming communications from the user menu of the scales:

The configuration that will be carried out as an example, will be that of a Master scales in section or department number 10, connected via Ethernet to other possible scales (with interconnected or floating vendors), and to the computer.

Access the menu by pressing the following keys:

↓ 9 4

Press the **V6** key 3 times (down arrow) to locate the OPERATIONAL function.

Press the **Vn** key (right arrow) to access the function.

Press the **V2** key (down arrow) to locate the COMMUNICATIONS function.

Press the **Vn** key (right arrow) to access the function.

The Communications configuration menu will be displayed. To modify any alphanumerical value or choice option (Yes/No), press the **Vn** (right arrow), to change any numerical value, press the numerical keys. To move up or down through the options, **V2** and **V6** can be used respectively.