Ditron s.r.l.

WinEcrCom DRIVER

List of revisions.

1	October 2001	First version
2	June 2002	First revision
3		
4		
5		
6		
7		
8		
9		
10		

2

CONTENTS

1. **Introduction**

The “WinEcrCom” driver is a set of programs allowing the connection of cash registers within the Windows environment.

The WinEcrCom driver creates an efficient “logic connection” between an application program and the ECRs connected to the PC. In fact the driver enables a set of functions which make it possible to make a connection, send commands to the ECRs, handle “events” produced by the ECRs and interrupt the connection.
The WinEcrCom driver enables full, physical connection to the device (whether directly using RS232 or in an RS485 network using the special adapter), and “hides” the connection protocol and the translation of all messages exchanged between the application program and the cash registers from the binary format used by the cash registers themselves, using immediately comprehensible, easy to use text format. In fact the WinEcrCom drive uses a special macro-language based on a simple set of instructions which makes it possible to send commands to the cash registers to take full advantage of all their functions. It is possible, for example, to print receipt slips from the PC, read the programming of the ECRs, program them, and transmit and receive files from the memory expansion card.

WinEcrCom is an instrument designed to be used by the people who develop management software in the world of retail applications. It must thus be understood as a set of “components” to be “integrated” within other programs and NOT as a self-standing program. WinEcrCom is particularly designed for users with a good knowledge of basic Windows programming who need to connect their applications to cash registers quickly and speedily.
WinEcrCom is based on the “COM” model. It is made up of various parts which will be described in detail below. Basically, it is divided into two parts; a “Service Object” and a “Control Object”. The “Service Object” is an executable program which is automatically activated every time an application program requests a connection to a cash register. The Service Object is fully responsible for the physical connection protocol and the translation of all messages exchanged. The “Control Object” is an ActiveX control; a component using standard technology which can easily be “inserted” into a Windows program written using any programming language which supports ActiveX technology (Visual Basic, Visual C, Delphi, Builder and many others). The Control Object makes a set of Methods, Properties and Events available to a developer which allow him to send commands to the cash registers in a very simple and rapid way.

Once the way the WinEcrCom driver works has been understood, and the macro-language it implements has been learned, all that will be required will be a few clicks of the mouse to add complete management and support for the connected cash registers to your own applications programs.

2. Terminology

In the rest of this manual, a few recurrent terms will be used in abbreviated form for simplicity. We give the full forms below:

- <u>Application</u>
  This is an executable program (or set of programs) realised by the developer for whom this manual is intended within which one wishes to integrate the possibility of connecting cash registers. Obviously the developer needs “sources” and must be able to

add the necessary parts of the code to include the functions made available by WinEcrCom.

- SO
  This is the “Service Object” or that part of WinEcrCom comprising the executable program “SoEcrCom.EXE”. The SO contains some interfaces which must be recorded in the system. This recording is normally carried out by the installation program.
- CO
  This is the “Control Object” or the ActiveX component, made up of the “CoEcrCom.OCX” which is incorporated within the Application. The CO must always be recorded in the system and this recording is normally carried out by the installation program.
- ECR
  This term stands for a connected cash register.

- WinEcrCom Language
  This is a series of commands of the “text” type which allow all the functions of the ECR to be recalled by the Application. It is very similar to a simple programming language in so far as it is made up of instructions and operands. The WinEcrCom language is directly derived from the one used by the interpreter of the DOS “ECROM.EXE” commands, with which it is almost fully compatible. The WinEcrCom language is defined by the “ECROM.INI” file, which, as described below, can also be modified.

3. Installing the WinEcrCom driver

In the WinEcrCom CD-ROM driver, there is a simple SetUp program which needs to be launched in order to carry out the driver installation process.
This program will extract and copy all the files necessary onto the PC’s hard disk. During installation, the program proposes the following directory:

“C:\Programs\WinEcrCom”

which can be confirmed or substituted by another name at will. This will be the base directory where all the necessary files will be copied and it will contain the following subdirectories and files:

- “DRIVERS” Subdirectory
  contains the driver files:

SoEcrCom.exe	(SO)
CoEcrCom.ocx	(CO)
EcrCom.ini	(language definition file)
EcrCom_i.ini	(Italian version of EcrCom.Ini)
EcrCom_e.ini	(English version of EcrCom.Ini)
Prot485.dll	(physical protocol handling for 485)
Prot232.dll	(physical protocol handling for 232)
EcrTrad.dll	(translator)
Modem.dll	(physical protocol handling for connection via Modem)

- “UTILITIES” Subdirectory
  It contains some utility programs, including:

WinEcrConf.exe	(Configuration utility for the logic ports)

The following two subdirectories are also in the CD-ROM:

- “DEMO” Subdirectory
  Contains various examples of driver use, subdivided into various programming languages, with their sources. It also contains a subdirectory called “EXAMPLES” with different kinds of files with the WinEcrCom commands.

- “DOC” Subdirectory
  Contains manuals and technical documentation.

The installation program updates the system and records various components. After installing and recording the driver components, the installation program automatically launches the logic port configuration program, described in the next section.

4. Driver configuration utility and definition of Logic Ports

In order to make correct use of the WinEcrCom driver, it must be configured properly and one or more "Logic Ports" must be defined.
A "logic port" defines how the ECRs will be connected up to the PC. For every logic port, it is necessary to configure some options to describe the connection.
The logic ports are numbered from 1 to N.
To set the options for WinEcrCom and create, modify or eliminate a logic port, the special "WinEcrConf.exe" program is needed. This program is launched automatically at the end of the installation procedure, but can also be used afterwards every time a change needs to be made to the driver configuration or an existing logic port, or else if other logic ports need to be defined.
At least one logic port needs to be created, otherwise the driver WinEcrCom cannot be opened.
In fact, as described later on, when an application has to open a new connection with connected ECRs, this is usually called the "Open" method, specifying which logic port needs using.
The "WinEcrConf.ex" asks the following information:

- Kind of physical connection. It may be :
  - RS232
    the direct connection of just one ECR to a serial port on the PC
  - RS485
    for the connection of up to 12 ECRs in a network using a 485 adapter installed onto a serial port on the PC
  - MODEM
    for the connection of a remote ECR using two modems.

- Serial Port
  Specifies the serial port used on the PC (COM1, COM2, COM3 or COM4).

- ECR protocol string
  Specifies the same protocol-string programmed on the ECRs. This string defines:
  - Communication speed (baud-rate)
  - Parity Bits (RS232 or MODEM only)
  - Number of stop-bits
  - Simplified protocol flag (No-Sync)
  - Long time-out flags
  - Checksum flag

  We recommend the value "60185" for RS232 and RS485.

- Network definition (RS485 only)
  This is a string defining which ECRs are connected in a 485 network. It is made up of a sequence of 12 characters which can be "0" or "1", where "1" stands for the ECRs present..
  The ECRS are numbered from 1 to 12, so it is necessary for the addresses to go from 1 to 12.
- Examples:
  Wanting to connect only 4 ECRs with the addresses 1,2,3, and 4, it is necessary to specify the value "111100000000".
  Wanting to connect the ECRs to the addresses 1,2,6,7,8,9 it is necessary to specify the value

"110001111000".

- On-line Option
  Activate this option only if ECR messages for access to external files and/ data-collect and/or interactivity are envisaged.

Furthermore, the program "WinEcrConf.exe", allows the configuration of the following general driver functions:

- "Auto-Run" Option
  Activate this option if the special Autorun function is to be used. It allows WinEcrCom commands to be run automatically just by creating the file, without any specific program.

"Auto-Run" Command file
When the AutoRun mode is activated, the "WinEcrConf.exe" program asks for the name (complete with research path) of a file which the driver will automatically execute as soon as that file is created and saved by any other program, for example using NotePad or another text-editor.
When the "Auto-Run" mode is active, the WinEcrCom driver, (i.e., the SO), continually controls (every second), the presence of the Auto-Run command file configured and as soon as it finds it, runs it. At the end, WinEcrCom removes the file from the hard-disk, writes the report file containing any errors and checks for the presence of the file every second.

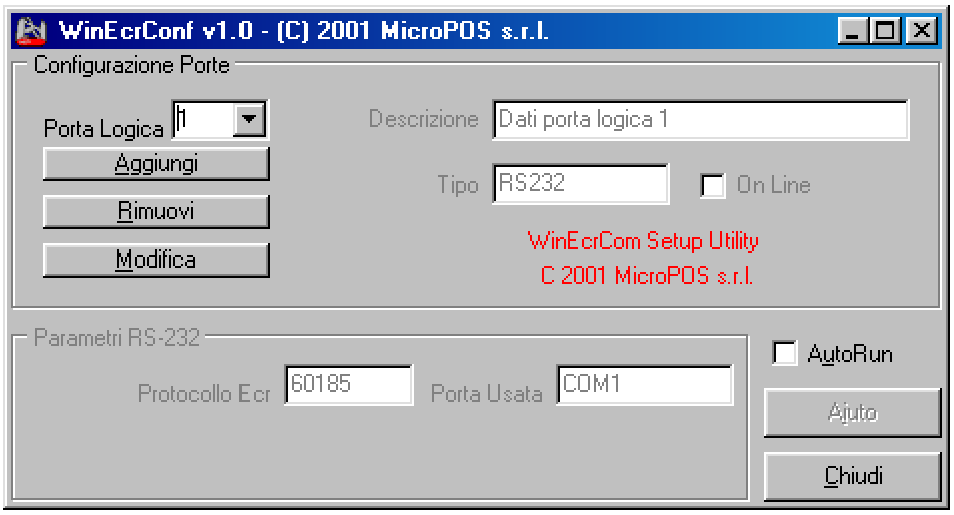
"Auto-Run" Error report file
When the "AutoRun" mode is activated, the "WinEcrConf.exe" program asks for the name (complete with research path) of a file which the driver will automatically write as a report of any errors occurring during the automatic running of the Auto-run command file.

4.1 Examples

Here follow some examples of driver configurations.

Example 1:

Logic Port:	1
Type:	RS232
Port used:	COM1
ECR protocol:	60185
On Line:	No
AutoRun:	No



WinEcrConf v1.0 - (C) 2001 MicroPOS s.r.l.

Configurazione Porte

Porta Logica: 1

Aggiungi | Rimuovi | Modifica

Descrizione: Dati porta logica 1

Tipo: RS232 | On Line

WinEcrCom Setup Utility
C 2001 MicroPOS s.r.l.

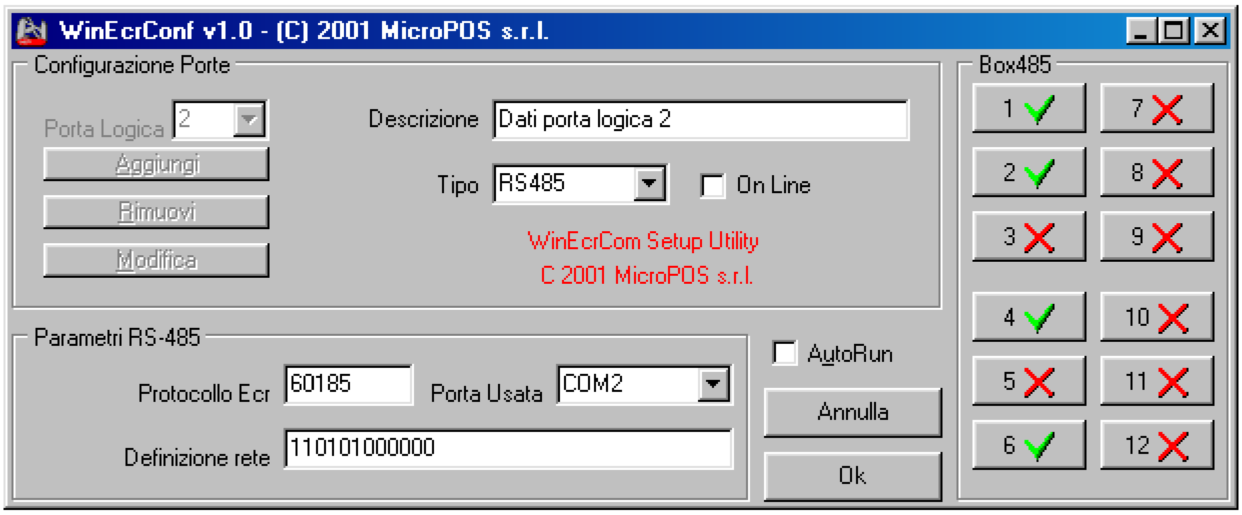
Parametri RS-232

Protocollo Ecr: 60185 | Porta Usata: COM1

AutoRun | Ajuto | Chiudi

Example 2:

Logic Port:	2
Type:	RS485
Port Used:	COM2
ECR Protocol:	60185
On Line:	No
AutoRun:	No
Network definition:	110101000000



WinEcrConf v1.0 - (C) 2001 MicroPOS s.r.l.

Configurazione Porte

Porta Logica: 2

Aggiungi | Rimuovi | Modifica

Descrizione: Dati porta logica 2

Tipo: RS485 | On Line

WinEcrCom Setup Utility
C 2001 MicroPOS s.r.l.

Parametri RS-485

Protocollo Ecr: 60185 | Porta Usata: COM2

Definizione rete: 110101000000

AutoRun | Annulla | Ok

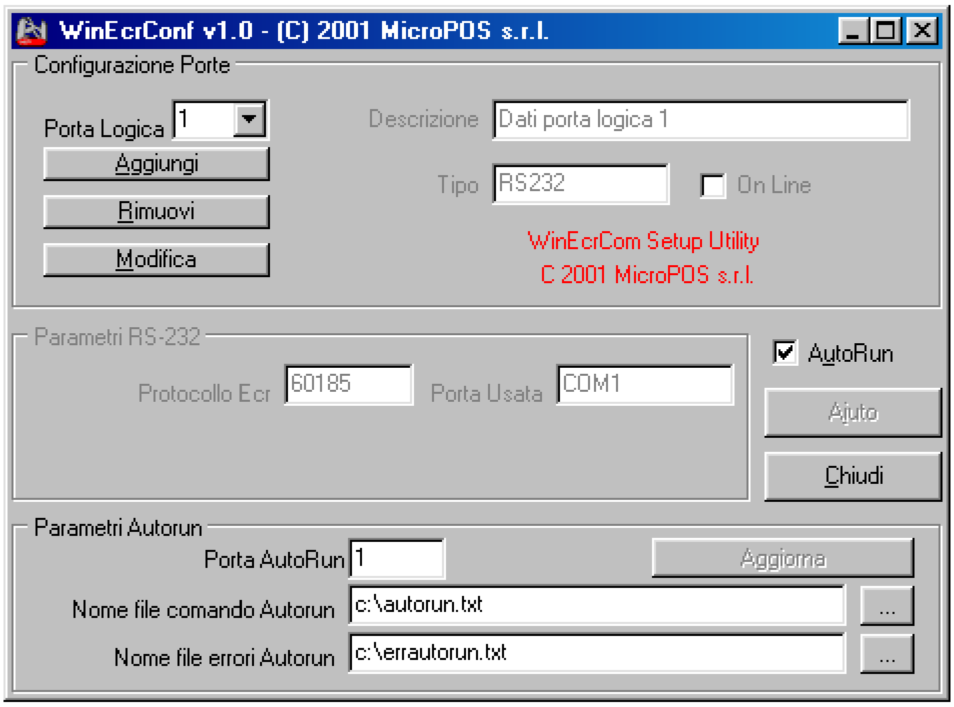
Box485

1 ✓	7 ✗
2 ✓	8 ✗
3 ✗	9 ✗
4 ✓	10 ✗
5 ✗	11 ✗
6 ✓	12 ✗

Example 3:

Logic Port:	1
Type:	RS232
Port used:	COM1
ECR protocol:	60185
On Line:	No
AutoRun:	Yes

AutoRun Port:	1
Autorun command file name:	c:\autorun.txt
Autorun error file name:	c:\errautorun.txt



WinEcrConf v1.0 - (C) 2001 MicroPOS s.r.l.

Configurazione Porte

Porta Logica: 1

Descrizione: Dati porta logica 1

Tipo: RS232

☐ On Line

WinEcrCom Setup Utility
C 2001 MicroPOS s.r.l.

Parametri RS-232

Protocollo Ecr: 60185

Porta Usata: COM1

☑ AutoRun

Parametri Autorun

Porta AutoRun: 1

Nome file comando Autorun: c:\autorun.txt

Nome file errori Autorun: c:\errautorun.txt

5. WinEcrCom language

The WinEcrCom driver defines a kind of programming language which allows commands to be sent to the ECRS in a simple and intuitive way.
The WinEcrCom macro-language is very similar to that adopted by the old DOS ECRCOM.EXE program, from which it is directly derived. It is possible to continue using the files used in the past to send commands to the cash registers using ECRCOM.EXE.

The new version of the language extends a few of the commands. The main addition is integration with the functions of the DOS EM_LINK.EXE program. In fact, a new “EM_LINK” instruction has been added via which it is possible to send and receive the archives supported by the INDIPOS memory expansion. With this new instruction it is possible to carry out all the operations which were previously carried out using the DOS EM_LINK.EXE.

The commands defined by the WinEcrCom macro-language can be sent directly to the ECRs via “methods” made available by the drivers, or else it is possible to create a text file in which one can write a sequence of several WinEcrCoM instructions, i.e. a “list of commands file” or “command-file”. This file is then passed to the driver, which scans the file, interprets and translates the commands and transmits them to the ECR to be carried out. This is in “batch” mode.

WinErCom’s CO makes the EcrCmd method available, allowing a single instruction or a whole file of instructions to be carried out. The WinEcrCom driver “interprets” the command sent and translates it into a series of messages which are transmitted to the ECR to carry out the specific instruction.

Each single command must be written on a single line of text (in ascii format) and must respect the syntax defined by the macro-language and described in this document.
It is possible to send the driver (using the EcrCmd method) a single instruction or a whole command file.
To send the driver a whole command file, you need to use the EcrCmd method, sending the method a line of text in this format:

"*@ name-file-command* [, *name-file-report-errors*]"

that is, a line which starts with the special character "@" followed by the name of the text file containing the list of commands to be carried out. It is also possible to specify the name of a file for any errors occurring while running. The name of the error-report file must be separated from that of the command file by a comma.

For every source line processed, the interpreter prepares a result message such as "OK" if the source line has been recognised correctly, translated and carried out by the ECR, or it can be an error message if the interpreter has found a syntax error in the source line or if the ECR gives off an error signal. This message is sent back by the EcrCmd function so that the application can verify the correct execution of the command.
In batch executions of a whole file, any errors generated by the execution of each single source line are queued in the error report file which may have been specified together with the name of the command file. For each error, the line number of the source will be indicated.
Using the special "Auto-Run" function, it is possible to configure the driver so that it will automatically carry out a specific file command as soon as the driver finds the file in the configured position. At the end of the execution, the source file is automatically cancelled. In this case it is not necessary to integrate the CO within an application or use the EcrCmd method to execute the file. It is sufficient simply to create a file with the commands to be sent to the cash registers and then copy or rename the file, with the name (and path) which has been configured as "Auto-Run Command File".
The driver itself will recognise the existence of the file and will carry it out automatically.

6. Command writing

The general structure of a "command" is as follows :

Instruction operand, operand, ... ;comment

Instruction :

This is an operative code which identifies the command itself or establishes the action to be taken by the ECR.
The interpreter transforms the lower case letters into upper case ones used for the instructions, i.e., there is no distinction between upper and lower case.

Operand :

This is generally an expression which defines the "value" of a "field".
Between *instruction* and the first *operand* there must be at least one separating space; the various *operands* must be separated by inserting a comma ",".

The operands can be entered in any order and other spaces can be inserted.
Generally, the operands are “optional” except in a few instructions which require compulsory ones.

Comment :
This is a free string which is not considered by the interpreter and which can be inserted into the document and comment on the line. The comment must be preceded by a semicolon “;”.

The general structure of the operand is like this :

Field name =value or simply
Field name

Field name :
This is a key-word which defines the field whose value is to be specified.
The interpreter transforms the lower case letters used for the field name into upper case so there is no distinction between upper and lower.

Value :
This is the value to be attributed to the specific field.
If the value is omitted, the specified value “TRUE” is attributed to the specified field, i.e., it is understood that the field is of the logic type, i.e. True or False.

Between the field-name and the value, there must be an equals sign “=”, possibly with a space to either side.

In general, the fields require a value which can be numerical or alphanumerical. If the value is numerical, this must be entered using numbers only and a decimal point required to specify the decimals. If the number is negative, a minus sign “-“ must be placed before the number.
The following are examples of numeric values :

```
100
25.5
-10
```

If the value is alphanumerical, this must be between inverted commas “” and must be made up of a string of ASCII characters..
If there is an apostrophe “’”, it must be preceded by another apostrophe “’”.
Examples of valid alphanumerical characters are:

```
‘PIZZA MARGHERITA’
‘CAFFE’’’
‘PAPA’’ E MAMMA’
```

The value may also be specified using a symbol, i.e. a label which is then automatically replaced by the numerical value (or string) assigned to the symbol used.

For example, some fields require the value “1” to indicate the field is set at “NO”, or the value “2” to show that it is set at “YES”. In this case it is possible to specify the symbol “YES” which is translated as “2” or the symbol “NO”, which is translated as “1”.

Example of a source line :

SALE DPT=3, PRICE=1500, DES=’COCACOLA’   ;Sale in dept 3

In this line, the instruction is “VEND” and three operands have been entered :

DPT = 3	*field-name* = “REP”	*value* = “3”
PRICE = 1500	*field-name* = “PRICE”	*value* = “1500”
DES = ‘COCACOLA’	*field-name* = “DES”	*value* = “’COCACOLA’”

A comment preceded by the symbol “;” has also been added

The *instructions* and the field-names may be entered also using more or fewer characters than generally expected.

6.1 Kinds of instructions

The following kinds of instructions are allowed:

- ECR control instructions
- Instructions for completing a receipt.
- Instructions for programming an ECR from a PC
- Instructions for reading data container in the ECR memory
- Special instruction “EM_LINK” to transmit and/or receive the supported archives from the memory expansion card.

7. ECRCOM.INI definition file

The syntax of the commands is defined in the file ECRCOM.INI which must be in the sub-directory "DRIVERS" of the installation directory of the WinEcrCom Driver. This file is divided into sections identified by a section-name between square brackets.
These files are freely modifiable and so it is possible to assign symbolic names which are different from the instructions and the field-names, as it is also possible to create "alternative" symbolic names called "aliases" so that, for example, the instruction "SALE" can be recognised by the interpreter also as "VEND" in Italian.

The first section [OPCOD] defines the various operating-codes in the language associating an order number to every symbolic name in the instructions. It is not possible to modify the order numbers of the various instructions but it is possible to modify the symbolic name.

Then come the sections which, for every instruction, define the operands for that instruction.
Consulting these files, it is clear which are the instructions implemented and which are the operands.
Then there are the sections for defining the symbolic labels; in particular the [EQUTASTI] defines the symbolic names for the code-keys of the ECR.
There are also some sections which define the error messages which the interpreter sends back and a section which defines the various text messages used by the driver. It is quite possible, then, to modify these messages and adapt them to different languages.
The installation program WinEcrCom automatically copies two different versions of ECRCOM.INI, i.e.:

- ECRCOM_I.INI = Italian Version
- ECRCOM_E.INI = English Version

The ECRCOM_I.INI is also copied with the name ECRCOM.INI, which is used by default. To use the English version just copy the file ECRCOM_E.INI into the file ECRCOM.INI.

The ECRCOM.INI file is supplied complete with comments describing all its parts.
Reading the file, we obtain information on the various commands implemented.
Furthermore, in the DEMO\EXAMPLES file, different examples of source files with the WinEcrCom commands are supplied which illustrated the majority of instructions implemented.

In this manual, only a few of the possible commands are described (including VEND and EM_LINK) as an example. Please read the file ECRCOM.INI and the examples supplied in DEMO\EXAMPLES for all other commands.

8. The “SALE” Instruction

This instruction activates a sale which can be either by “department” or “PLU”. The full syntax is:

SALE DPT=*department-number*, PLU=*article code*, PRICE=*price*, DES=*description*,
QTY=*number of articles*, No sale, VOID

DPT	specifies the number of the department if by “department”
PLU	specifies the article code
PRICE	specifies the unit sale price; if it is omitted, the ECR applies the price programmed for that department/article.
QTY	specifies the quantity, i.e. number of articles to be sold; if omitted, the ECR works on single item quantities.
DES	specifies the symbolic description to print on the receipt, i.e. dept/art names. If omitted, the ECR prints the description programmed for the dept/art.
REFUND	if present, specifies returned goods, i.e. it makes the sale negative.
VOID	if present, specifies that the sale must be voided, cancelling a previous sale.

9. The “FUNCTION” (General Function) instruction

With this instruction it is possible to send a general command to an ECR in “Key emulation” and so it is possible to use these instructions to carry out any function of the ECR.

The “FUNCTION” instruction is of the following type:

FUNCTION NUMBER = *numerical input*, ALFA = *alphanumerical input*, TERM = *function code*

NUMBER = *numerical input*
Specifies the possible numerical value of the input

ALFA = *alphanumerical input*
Specifies the possible alphanumerical value of the input

TERM = *function code*
Specifies the code of the required function.

In general the ECR accepts a numerical value for the input and a function key (also called “terminator”). For a few functions there is an “alphanumerical function”.
Every “terminator” is identified by a function code. For example the “Department 1” function has the code 75.

Thus, for example, to call up a sale of €10 in dept 1, just type in the following instruction
:

FUNCTION NUM=10, TERM = 75.

In the ECRCOM.INI, the [EQUTASTI] paragraph defines the symbolic values for all the function keys on the ECR (Labels). In this way, the instruction can also be written as follows:

```
FUNCTION NUM=10, TERM = DPT1
```

The interpreter of the WinEcrCom commands substitutes the value 75 for the symbolic constant DPT1 and then carries out the instruction.

10. The EM_LINK Instruction

This instruction uploads files to the memory expansion card in the cash registers and downloads them from the card.
The data format is the same as that used by the EM_LINK.EXE for DOS program.

Before using the EM_LINK command, it is necessary to send the "SELEZ ECR=n" command to select the ECR the transfer is going to.

EM_LINK CMD = *command*, FILE = *input/output data files,* ERR=*error files,*
RESET, CLEAR, BLOCK, APPEND, INFO,
SELECT = *article selection criteria*, SELDPT = *dept no,*
NUMBER = *receipt number*, DATE = *receipt date,*
TRANSLATE, OFFLINE

Here follows a description of the operands.

CMD: alphanumerical
Shows the command code, which may be:

'UA' = Upload Articles	Sends the article file to the ECR
'DA' = Download Articles	Receives article files from ECR
'UO' = Upload Offers	Sends the offer file to the ECR
'DO' = Download Offers	Receives article files from ECR
'DM' = Download Operations	Receives the data-collection file Y (operations)
'RM' = Zeros operations	
'DT' = Download GT Totals	Receives the file with the Grand Totals
'DF' = Download Daily fiscal totals	Receives the daily fiscal totals files

FILE : alphanumerical
Specifies the full names of data files.
These names show the file containing data to be transferred in the case of an up-load command ('UA', 'UO').
They show the file name where the received data will be stored in the case of a download ('DA', 'DO', 'DM', 'DT', 'DF').

ERR : alphanumerical
Specifies the full name of a file where any errors occurring during the process will be stored.

RESET
This option requires the total cancellation of the article archive when a UA type command is given.

CLEAR

This option zeros the “Sales Quantity” and/or “Sales Value” fields when a “DA” command is given.

BLOCK [= n]

Temporarily blocks the ECR while a command is being carried out. It is possible to specify an optional numerical value, which can be:

BLOCK = 1	:	ECR block (the same as just BLOCK)
BLOCK = 2	:	if the transaction is in progress at the ECR (during sales), the driver waits until it is finished, and only then blocks the ECR and carries out the command. During the wait, the driver sends the “Progress” event continuously, with an appropriate message allowing the application to abort the operation.

APPEND

This option opens data output files in “append” mode, i.e. queuing the data which has just been read onto the contents of the specified file

When the APPEND option is not selected, the specified data file is recreated anyway, writing over any data contained in it.

INFO

Selecting this option, the driver adds a few command lines (starting with “,”) to the exit file for downloading. These comments contain information on the command requested, the address of the ECR in the network, its serial number and version, the date, time and result of the operation.

SELECT = n

This value determines selection criteria for article records obtained using the “FROM” command. The value “n” may be:

0 = get all fields of all records present in the archive
1 = get only the Code, Stock, Qty, and Value of all records
2= get all the values of transaction records
4 = get only the Code, Stock, Qty, and Value of all transaction records

SELDPT = r

This value determines a selection criterion for article records based on the department. The value “r” shows the department number for getting the articles.

NUMBER = n

This value shows a selection criterion for data-collection records (“movements” archive). It allows only transactions referring to a specific receipt number to be read (without removing them).

Selecting NUMSCO = 0, all the transactions present in the memory will by read, without removing them, however.

DATE = ‘DD MM YY’ (alphanumerical)

With this operand, it is possible to specify also the date “DDMYY” for data extraction (data collection). NOTE; in this case, the records extracted will not be removed from the ECR memory.

TRANSLATE

This option required the “translation” of transaction records (data-collection) from the internal format of the ECR, to the format exported by the WinEcrCom driver for data collection messages Y (see paragraph “Translation of data-collection messages”).

OFFLINE

This option can be used for 485 network connection of more than one ECR, when the On-Line option is activated.
Selecting the OFF-LINE option, the driver “suspends” the continuous scanning of all the cash registers connected, to increase the speed at which the command is carried out, i.e., the On-Line mode is temporarily suspended.

The data format used for the Articles and Offers archives is the same as the one used in the old DOS EM_LINK.EXE. It is therefore possible to use the same files. As for output files, it is possible to modify some alignment criteria using the “OutEditOptions” property (see the paragraph “Properties”)

Examples :
The DEMO\EXAMPLES subdirectory, in the WinEcrCom installation disk, also contains a few examples of the use of the EM_LINK instructions.

11. Instructions for program files

WinEcrCom makes some instructions available for programming the PC. The cash register has some programmable data files. Typically there are the following files:
1 = Departments
2 = PLU
3 = VAT
4 = Heading
5 = Modifiers (percentages A and B)
6 = Options
7 = Foreign currency
8 = Groups
9 = Subtenders

For each of these files, there is a specific instruction and the operands for these instructions define the values to be programmed in the various fields. Programming takes place using keyboard emulation, i.e., the command interpreter translates instructions into the relative key sequences normally envisaged for manual programming acting on the cash register keyboard. This allows the exact simulation of programming as if on the ECR keyboard.

The operands of the program files instructions are generally optional: if there is a non specified field, the interpreter creates a sequence which does not modify that field, i.e., it leaves the current value, otherwise, it changes it. Given the simulation of the manual programming sequence, the cash register can print the normal programming receipt and so there is a paper record of the programming carried out. The print option can be disabled.

11.1 The “STARTSET” instruction

This instruction is used to activate the “file program” mode.
It puts the ECR into SET mode and makes the interpreter carry out its file programming instructions on the ECR, until the special instruction “ENDSET” which ends the “file program” mode.

The full syntax is :

```
STARTSET  NOPRINT
```

The NOPRINT operand is optional and is logic, i.e. it is not necessary to input any values.
If NOPRINT is selected, the ECR will not print anything during the file program instructions. If it is not specified, the ECR will print normal receipts summarising the programming carried out.

NOTE : The STARTSET instruction must be entered before the first file programming instruction for the ECR

11.2 The “ENDSET” Instruction

This instruction ends the special file programming mode on the cash register.

There are no operands
NOTE : The **ENDSET** instruction must be entered after the last program files on the ECR.

11.3 The “SETDPT” instruction

This instruction is for programming a department or should be used only when the program files option has been selected on the ECR.

The complete syntax is :

SETDPT NR=*department number*, DES=*description*, PRICE=*price*, LIS=*max number of figures*, VAt=*VAT code*, SI=*single stroke flag*, GROUP=*group code*

NR	specifies the number of the department and is compulsory
DES	specifies the alphanumeric code to give to the department
PRICE	specifies the basic price value for the department
LIS	specifies the listing-capacity i.e. the maximum number of figures allowed for input in that department
VAT	specifies the code for the VAT rate to be applied to the department
SI	selects the single stroke option for that department and the value to be assigned must be 1 or NO, if not required, or 2 or YES, if required.
GROUP	specifies the group code

12. The “READ” (File reading from ECR) instruction

This instruction allows files to be read from the ECR. The syntax is:

READ FN=*file-number*, FILE=*file-name*, SETPROG, APPEND

The descriptions of the operands :

NF	Number of file to be read
FILE	Name of file to place read files
APPEND	Instruction to queue data
SETPROG	Includes the lines "PROG" / "ENDPROG"

13. Translation of Data-Collection Y (Operations) messages

The data collection Y messages are transmitted by connected ECRs every time any function is carried out or transactions are memorised in the “transactions” file present in the memory expansion file and in this case, they can be accessed using a command of this kind:
 “EM_LINK CMD=’DM’, FILE=….. “ (download transactions).

In any case, the data-collection records can be translated by WinEcrCom so that they can be exported in as independent a format as possible from the particular ECR they were generated by and which makes the analysis by a generic application program much easier when extracting data regarding sales, whatever level of detail is required.

For every record translated, a text line is created with a series of fields separated by the set separating character, which is the comma by default.
The first field is a function code, i.e. a numerical value which identifies the specific function carried out by the ECR. The fields which follow on depend on the function-code. There can be numerical and alphanumerical codes, which must be between inverted commas.
The numerical fields can be a symbol, as in the case of amounts and quantities, or without, as in the case of article codes or operator number.
The numerical values are always given without putting in the decimal point even for fields requiring decimals. Use the fixed comma system.

For example, all amounts must always be considered with the fixed number of decimal places (e.g. two in the case of Euro prices) and the quantity of objects sold must be considered to three places.

The format of numerical fields depends on the value set for the OutEditOptions properties. Refer to the appropriate paragraph.

The function code is always shown by the first two characters of the line. It can go from “01” to “99”.

Here is a brief description of the function-codes used. A few of the fields are “optional”, i.e., they can be omitted, and are shown between square brackets in the following text. “ND” stands for the number of decimals of the ECR’s basic currency.

<u>02 : Sale by dept</u>

02, dept, qty, value

dep	Department number
qty	Quantity sold, with symbol and three decimal places
value	Total sales, with symbol and fixed ND.

<u>03 : Sale by article</u>

03, code, dept, qty, value [,offer code, discount]

code	code for the article sold; can be a number or alphanumerical. If so it must be between inverted commas.
dept	Department of origin
qty	Quantity sold to three decimal places
value	Total sale value, with symbol and number of decimal places fixed (ND)
cod.off.	any special offer code on the sale item
discount	Value of any discount applied to the sale of an item on offer. Field with symbol and fixed Decimal Places (ND)

04 : Closure of transaction

04, time-stamp, num.rec, total, [stamps]

time-stamp	date and time string (alphanumeric) in ‘DD/MM/YY HH:MM’ format
num.rec	Number of issued receipt (issued)
total	Total of receipt, without symbol and with ND fixed decimals
stamps	The number of stamps given if any

05 : Non Add Key

05, number

number	The number which has been typed by the cashier

06 : Change cashier

06, cod.cas

cod.cas	numeric code (number) of the operator which has been entered and approved

07 : Payment (given value)

07, tender, value[,tender, value] [,tender, value]….

tender	Code for payment type (tender-code) selected
value	Amount given

08 : Change mode key

08, num.key

num.k	Number of operating mode selected

<u>09 : discount/surcharge % on the item</u>

09, perc, value, [code]

perc	Value of the percentage applied, to two fixed decimal places
value	Amount of discount/surcharge, with symbol and ND fixed decimals
code	Any article code if a discount has been applied to an article. It may be numerical or alphanumerical.

<u>10 : Discount/% surcharge on Subtotal</u>

<u>11 : Absolute Discount on item</u>

11, value, [code]

value	Amount of the discount, with symbol and ND fixed decimal places
code	Possible article code if a discount has been applied to an article. It may be numerical or alphanumerical

<u>12 : Absolute Discount on Subtotal</u>

12, value

value	Amount of discount, with symbol and ND decimal places

<u>14 : Subtotal</u>

<u>16 : Closure of transaction on slip-printer</u>

<u>17 : Chip-card Enquiry</u>

18 : Chip-card Update

19 : Set List

20 : Offer

21 : Absolute increase on item

21, value, [code]

value	Amount of increase, with symbol and fixed decimals
code	Any code if a discount has been applied to an article. It can be numerical or alphanumerical

22 : Absolute increase on Subtotal

22, value

value	Amount of discount, with symbol and fixed decimal places

30 : “Entrance Ticket” special discount

14. “CoEcrCom” (CO)

The Control Object (CO) made available by the WinEcrCom driver, is the ActiveX object which can easily be incorporated into a generic application program, so as to establish an efficient logic connection between the application and the connected ECRs.

The “CoEcrCom.ocx” file contains the object to be included in the applications. This file is automatically “recorded” in the WinEcrCom installation program system. Like all ActiveX objects, the CO makes a set of Methods, Properties and Events to the programmer.

Here follows a brief description of these Methods, Properties and Events. In the DEMO subdirectory of the driver installation disc, there are some examples of VisualBasic programming and C++ used in the CO, to which reference should be made for a more detailed and practical description.

15. Error codes

Any errors found by WinEcrCom while carrying out an operation are listed in the error files, (default ERR.OUT).
They appear in summary on screen, at the end of the operation. The type of error is shown using the following classification:

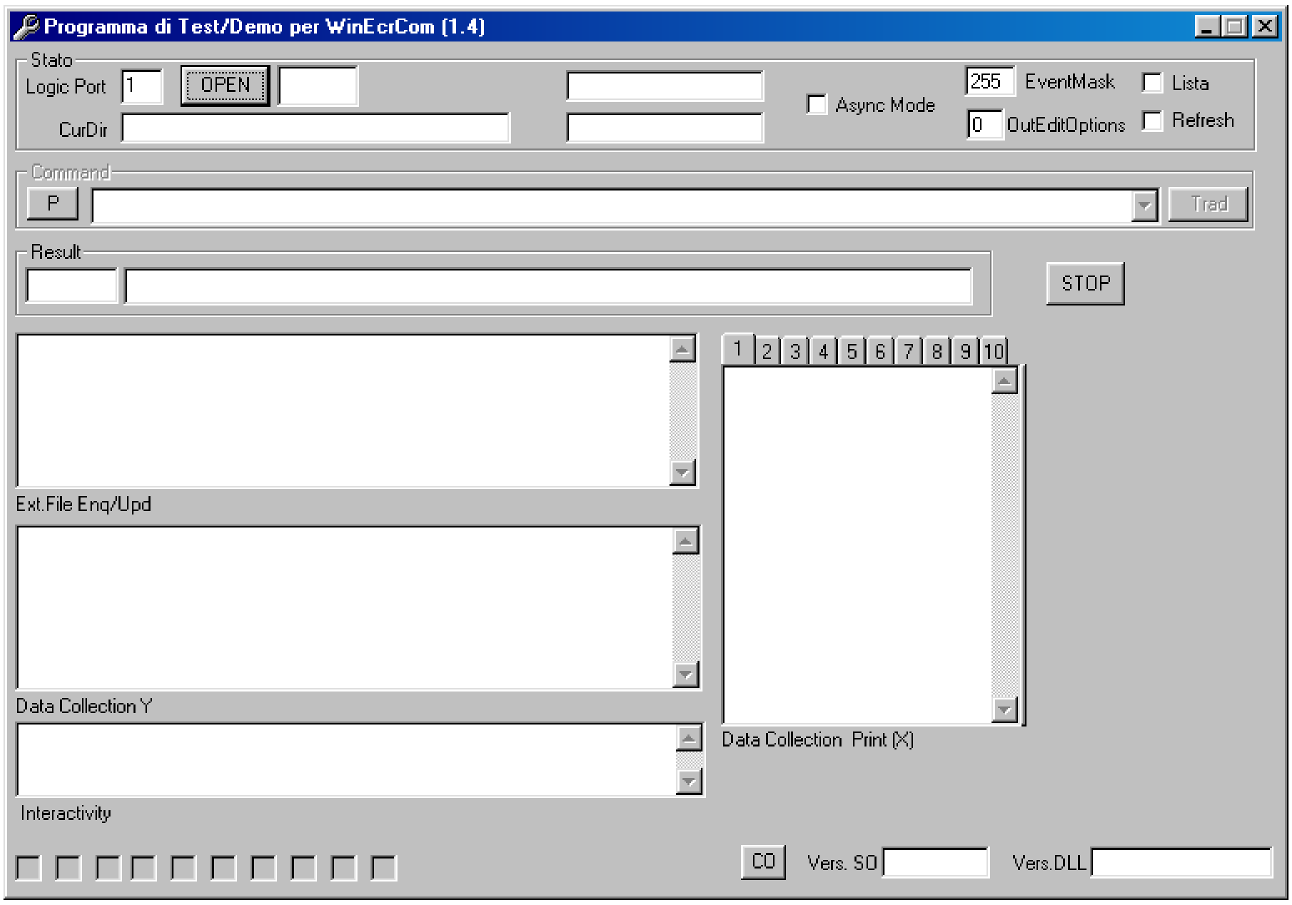
TYPE	CODE	DESCRIPTION
0 (BASE)	1	Driver off (SO not been activated)
	2	Translator not initialised
	3	Programming session in progress
	4	Impossible to read ECR configuration
	5	ECR with incompatibile version
	6	ECR not in Idle mode
	7	Programming session not under way
	8	ECR not in REG mode
	9	Memory expansion card not initialised
	10	Invalid batch line
	11	Input file failure
	12	Error file failure
	13	Log file failure
	14	Batch error (ECRCOM file)
	15	Transaction not under way
	16	Execution error
	17	SO error (driver not found or not registered)
	18	Driver already running
	19	Driver busy (command already in execution)
	20	Logic port not available or configured
	21	Property setting error
	22	Invalid parameters in the OPEN string
	23	Invalid properties (SETP instruction error)
	24	Invalid properties (GETP instruction error)
	25	chip-card Errore
	26	Error chip-card not blank
	27	No ECR selected
	28	Modem connection Time-out
	29	Modem error
	30	Broadcast transmission (SPECIAL) error
1 (SYNTAX)	0	Syntax error
	1	Instruction not found
	2	Instruction ambiguous
	3	Instruction illegal
	4	Operand not found
	5	Operando ambiguous
	6	Operand illegal
	7	Constant not found
	8	Invalid alphabetic Value (inverted commas)

	9	Illegal value
	10	Compulsory operand missing
	11	Sudden line end
	12	Instruction not accepted by this translator
	13	DEPARTMENT or ARTICLE missing on VEND
	14	Department number not allowed
	15	File number not valid for READ
	16	File number on READ not accepted by ECR
	17	ECR does not support groups
	18	Source line too long
	19	Illegal ECR number
	20	Cashier number not allowed
2 (I/O ERROR)	0	I/O Error (both for ECRCOM and EM_LINK commands)
3 (FATAL ERROR)	0	ECR indicates fatal error
4 (DATA ERROR)	0	ECR indicates data-error
5 (ECR ERRORS)	0	ECR indicates errore
6 (BASIC ERROR IN CARRYING OUT EM_LINK COMMAND	0	EM_LINK error
	1	Errors have occurred while running EM_LINK
	2	Error in EM_LINK command or command not allowed
	3	Error in opening input file
	4	Error in opening error file
	5	Error in opening log file
	6	Error while zeroing article file
	7	Impossible to activate ECR stand-by
	8	Error in opening “Download Articles” session on ECR
	9	Sudden closure of “Download Articles”
	10	Error in #records totals received in download check
	11	Operations file in overflow
	12	Impossible to activate block. ECR transaction in progress
7 (SYNTAX ERRORS CARRYING OUT EM_LINK COMMAND)	0	Syntax errors
	1	Sudden line end
	2	Alphabetical value not valid (inverted commas)
	3	Article code not valid
	4	Non-numerical value
	5	Invalid VAT value
	6	Invaliddepartment
	7	Invalid command
	8	Numerical value not allowed
8 (ECR ERROR IN CARRYING OUT EM_LINK	0	ECR indicates an error

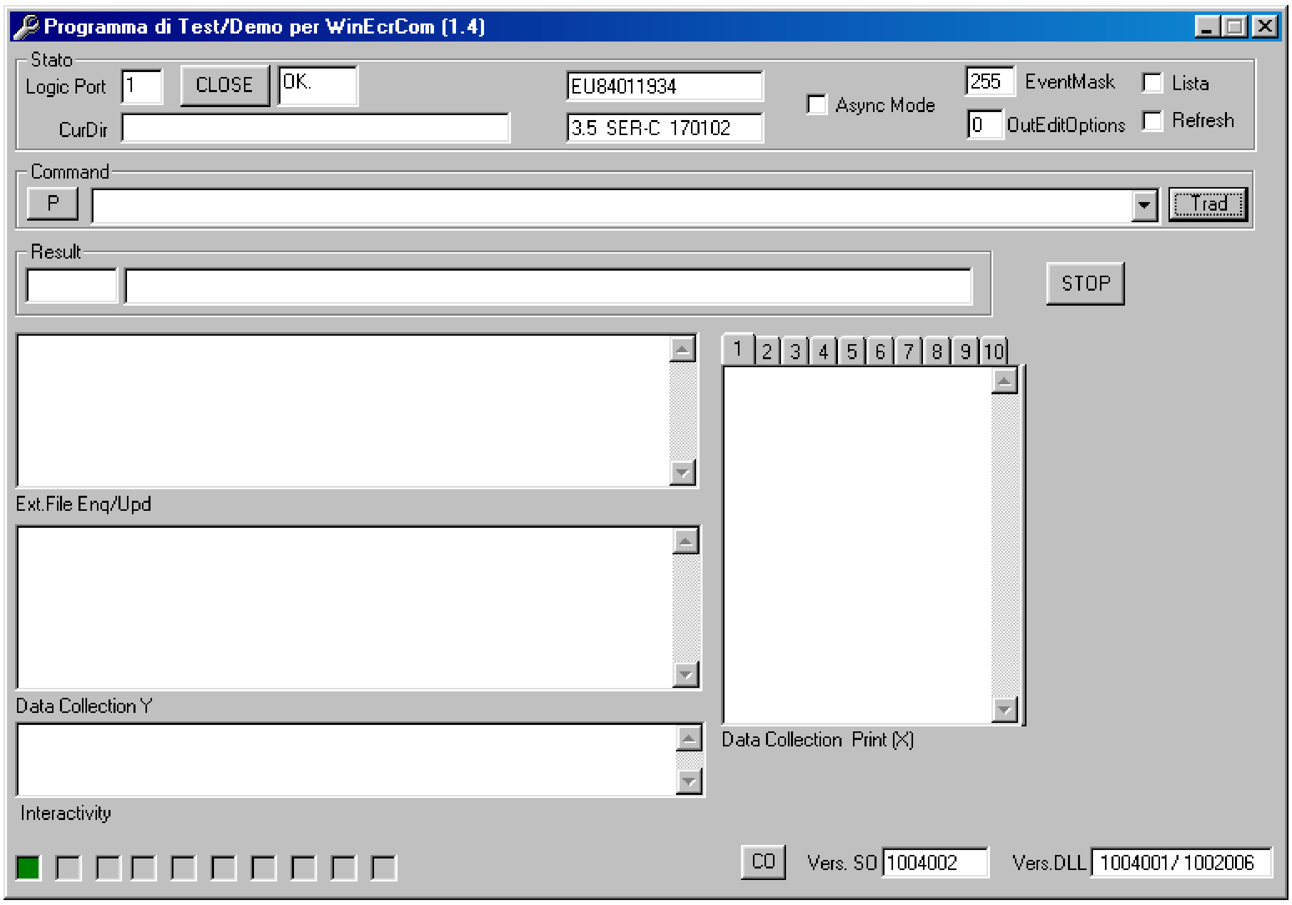
COMAND)		
9 (WRONG RESPONSE ERROR)	0	Wrong response error

16. Example of the use of the “Test/Demo Program for WinEcrCom

Within the WinEcrCom\Demo\Vb, it is possible to find the “DemoVB.exe” executable file. It is a test program which allows driver checks to be run. Here is the main screen:



If you enter the Logic Port value for the port you want to use in **“Logic Port”** (e.g. 1), and click on “**Open**”, the Service Object will go into operation. If the connection is ok, this will be the result:



and next to the “**Close**” key, will appear “OK” and the serial number of the cash register (e.g. EU84011934) and the firmware version (e.g. 3.5 SER-C 1701702). Here is a description of the various parts of the screen.

CurDir This is where the current directory must be entered. If you don’t write anything, by default, the directory WinEcrCom\Demo\Vb will be used.

N.B. To go into the desired directory, first enter the CurDir and then click on “Open”

Async Mode By setting this flag, the asynchronous mode is activated.

EventMask. This has a value of from 0 to 255.

OutEditOptions See the “Properties” section.

Lista Selecting this flag will bring up a list showing the command lines in progress

Refresh

Command Here, you insert the commands to be carried out.
Clicking on “P” brings up a series of preset commands.

Trad Clicking on this button, the commands in “Command” are put into operation.

Result In this window, the result of the operation will come through.

Stop Blocks commands put into operation.

Ext.File Enq/Upd Sales data are visualised (if “External Files” has been activated on the ECR).

Data Collection Y All “Data Collect” messages are visualised (if they have been activated on the ECR).

Interactivity **All “Interactivity” messages will be shown** (if they have been activated on the ECR).

Data Collection Print (X) All lines printed on the receipt will be visualised (if “Print Monitor has been activated on the ECR).

CO Brings up the Control Object version.

Vers. SO Visualises the Service Object version.

Vers. DLL Visualises the DLL version.

17. Output Model

The WinEcrCom model is of two types:
synchronous and asynchronous. A given class of devices can support one type, both types or neither type.

<u>**Synchronous Mode**</u>

This mode is preferable when a command can be carried out fast. Its advantage is its simplicity. The application calls the EcrCmd method which works only when the command has been fully carried out.

<u>**Asynchronous Mode**</u>

This mode is preferable when a command requires slow hardware interaction. The application calls the EcrCmd which works immediately, i.e. the command is carried out by the driver in background, and in the meantime, the application can process another code. When the command has been carried out, the driver sends a “CommandComplete” event, to inform the application of the carried out action and the result of the operation. In this way, it is advisable to make sure “CommandComplete” is on.
The asynchronous mode runs on a first-in first-out basis.

18. Properties, Method and Events

Properties

Name		*Access*	*Type*
ControlObjectDescription	1.0	String	R
ControlObjectVersion	1.0	Long	R
EnableTradDC	1.0	Bool	
EventMask	1.0	Long	
OperatingMode	1.0	Long	
OutEditOptions	1.0	Long	
QuoteChar	1.0	Long	
ResultCode	1.0	Long	R
SeparatorChar	1.0	Long	
ServiceObjectVersion	1.0	Long	R
Status	1.0	Long	

Methods

Name		
AboutBox	1.0	Void
Close	1.0	Long
DirectIO	1.0	Long
EcrStatus	1.0	Long
EcrStatusEx	1.0	Long
EcrCmd	1.0	Long
EcrConfStr	1.0	String
EcrConfVal	1.0	Long
Open	1.0	Long

Events

Name		
CommandComplete	1.0	Void
DataX	1.0	Void
DataY	1.0	Void
FileExtEnq	1.0	Void
FileExtUpd	1.0	Void
ProgressCtrl	1.0	Void
StatusChange	1.0	

18.1 Properties

ControlObjectDescription Property

Syntax **BSTR ControlObjectDescription;**

Observations String identifying the Control Object and the firm which produces it.
The property identifies the Control Object. An example of a returned string is:
"CoEcrCom Control, Copyright(C) 2001 Ditron"
This property is always visible.

See also **ControlObjectVersion** Property

ControlObjectVersion Property

Syntax **LONG ControlObjectVersion;**

Observations Control Object version number.

This number maintains the version number of the Control Object. Three version levels are specified, as follows:

Version level	Description
Major	"millions". A change to a superior version of WinEcrCom for a class of devices reflects significant interface improvements and can remove support for old interfaces from preceding superior levels
Minor	"thousands". A change to a lower WinEcrCom version level for a class of devices reflects significant interface improvements and must provide a set of preceding interfaces at that later version level.
Release	"units". Internal level supplied by the Control Object supplier. Updated when corrections are made to CO implementation.

An example of a version number is:
1002038,
This value can be shown as version "1.2.38" and interpreted as greater version 1, lesser version 2, release 38 of control object.
These properties can always be read.

See also **ControlObjectDescription** Property

Note
A Control Object for one class of devices operates with all Service Objects of the same class while ever the superior version number corresponds to the superior version number of the Service Object. If they correspond, but the number of the inferior version of the Control Object is greater than the inferior number of the Service Object, the Control Object can support a few new methods or properties, which are not supported by the Service Object release.
The following rules are applied to the API supported by the Control Object release, but not supported by the older Service Object release.

- Upon reading a non-supported property: the Control Object returns the non-initialised value of the property
- Writing a non-supported property: the Control Object returns, but has to remember that a non-supported property writes, or a method call has occurred. If the application reads the **ResultCode** property, the Control Object must return a WEC_E_NOSERVICE value (rather than read the current **ResultCode** from the Service Object). It must do this until the next property write or method call, at which time ResultCode is set by that API.
- Upon calling a non-supported method: the Control Object returns a WEC_E_NOSERVICE value and must remember that a non-supported property writes or a method call has occurred
- If the application reads the **ResultCode** property, the Control Object must return a WEC_E_NOSERVICE value (rather than read the current **ResultCode** from the Service Object). It must do this until the next property write or method call, at which time ResultCode is set by that API.

ResultCode Property

Syntax **LONG ResultCode;**

Observations This property is set from all methods. It is also set when a writeable property is set.

This property can always be read. Before the **Open** method is called, it brings up the value WEC_E_CLOSED.

The result code values are:

Value	Meaning
WEC_SUCCESS	Operation successful.
WEC_E_CLOSED	You have tried to contact a non operational device.
WEC_E_NOSERVICE	Control cannot communicate with Service Object. It is likely that a setup or a configuration error must be corrected
WEC_E_DISABLED	The operation cannot be carried out while the device is non operational.
WEC_E_ILLEGAL	You have attempted an illegal or impossible operation or an invalid parameter has been used.
WEC_E_NOHARDWARE	The device is not connected to the system or is not connected to the power supply.
WEC_E_OFFLINE	The device is off-line.
WEC_E_NOEXIST	The file name (or other specified value) doesn't exist.
WEC_E_EXISTS	The file name (or other specified value) already exists.
WEC_E_FAILURE	The device is unable to carry out the request even if connected, plugged in to power source and on-line.
WEC_E_TIMEOUT	The Service Object is in time-out awaiting response from device, or Control has gone into time-out awaiting a response from Service Object.
WEC_E_BUSY	The Service Object cannot allow request at the moment. E.g., if asynchronous output is in operation, some methods may not be possible.
WEC_E_EXTENDED	There has been a specific class of error conditions. The code for the error condition is available in the **ResultCodeExtended** property

See also "Status"

ServiceObjectVersion Property

Syntax **LONG ServiceObjectVersion;**

Observations Service Object version number.
This property maintains the number and version of the Service Object. Three levels of the version are available, as follows

Version Level	Description
Major	"millions". A change at superior WinEcrCom version level for a device class reflects significant interface improvements and can remove support for old interfaces from previous levels of superior versions.
Minor	"thousands" A change to a lower WinEcrCom version level for a class of devices reflects significant interface improvements and must provide a set of preceding interfaces at that later version level.
Release	Units Internal level supplied by the Service Object supplier. Updated when corrections are made to SO implementation.

An example of a version number is:
    1002038
This value can be visualised as version "1.2.38" and interpreted as a superior version 1, inferior version 2, release 38 of the Service Object.
This property is visualised from the **Open** method.

Note
A Service Object for a class of devices will operate with a Control Object for that class, until its superior version number meets the Control Object version number. If they meet, but the inferior version number of the Service Object is higher than the Control Object's lower number, the Service Object can support some of those methods or properties which cannot be accessed from the Control Object release. If the application requires these characteristics, it will have to be updated to use an older version of the Control Object.

EnableTradDC Property

Syntax **BOOL EnableTradDC;**

Observations This property allows data-collection Y messages to be translated.

EventMask Property

Syntax **LONG EventMask**

Observations The events can be activated or not depending on the properties settings. Their value is a number deriving from the sum of the various “weights”. Every “weight” indicates a particular event, according to the following table:

Data Collect X :	weight 1 (bit 0)
Data Collect Y :	weight 2 (bit 1)
External PLU Enquiry :	weight 4 (bit 2)
External PLU Update :	weight 8 (bit 3)
Interactivity :	weight 16 (bit 4)
Progress :	weight 32 (bit 5)
Command Complete :	weight 64 (bit 6)
StatusChange :	weight 128 (bit 7)

If you require all weights, then :
1+2+4+8+16+32+64+128 = 255

OperatingMode Property

Syntax **LONG OperatingMode**

Observations This property influences the way the EcrCmd method works, i.e. the method needed to send commands to the driver.

If OperatingMode = 0 => “synchronous” i.e. the EcrCmd method comes straight back, that is, the command is carried out by the driver in background and, in the meantime, the application can handle other codes. When the command is complete, the driver will send a “Command Complete” event to report to the application that the operation has been carried out. In this mode, it is advisable to have CommandComplete activated.

OutEditOptions Property

Syntax **LONG OutEditOptions;**

Observations This property controls the way in which the output files will be edited. The current possibilities are:

0	:	Lines up the fields and inserts separator characters
1	:	Lines up the fields, but does not insert the separator characters
2	:	Does not line up the fields, but inserts separator characters

QuoteChar Property

Syntax **LONG QuoteChar;**

Observations This property allows you to change the “inverted comma” character (e.g. present in the description of the article record) into another character..

SeparatorChar Property

Syntax **LONG SeparatorChar;**

Observations This property allows you to change the separation “comma” between the various fields (e.g. vend rep=1, pre=1) into another character.

Status Property

Syntax **LONG Status;**

Observations This property reports the state of the Service Object. The following states are possible:

State	Meaning
IDLE	Waiting (ready to carry out EcrCmd command)
BUSY	Carrying out an asynchronous command
CLOSED	**Open** has not been carried out
ERROR	An error has occurred

18.2 Events

CommandComplete Event

Syntax **VOID CommandComplete (LONG** EcrNum, **BSTR** ResultString,
LONG ResultCode**);**

Observation if a command is carried out in asynchronous mode, the event sent by the SO is handled when the command has been fully carried out.

DataX Event

Syntax **VOID DataX (LONG** EcrNum, **BSTR** vString**);**

Observations A data-collect X message has been received. The received message is visualised in the channel's TextBox.

DataY Event

Syntax **VOID DataY (LONG** EcrNum, **BSTR** vString**);**

Observations A data-collect Y message has been received.

FileExtEnq Event

Syntax **VOID FileExtEnq (LONG** EcrNum, **LONG** ArtType, **BSTR** ArtCode,
DOUBLE Qty, **LONG** *pRecType, **BSTR** *pRecord**) ;**

Observations An External File Enquiry message has been received. This happens when an ECR sends a message to request article sales data. The routine for this event basically means entering a database and preparing a "response" with the article data.

Input Parameters, i.e. those passed on by the SO :

EcrNum = The number of the ECR which made the request
ArtType = Type of article code:
0 = Numerical
1 = Alphanumerical
ArtCode = String with the code of the article to be found
Qty = Quantity the ECR is to sell

Output parameters, i.e. what the routine MUST return to the SO.

pRecType = Record Type. Tells the SO the type of record it is supplied with

0 = “normal” type
1 = “with special offer ” type

pRecord = String with “article record”
For type 0, this must be made up of the following, separated by a comma:
<code>,<description>,<department>,<offer code>, <conditions>, <price 1>,<price 2>,<price 3>

FileExtUpd Event

Syntax **VOID FileExtUpd (LONG** EcrNum, **LONG** ArtType, **BSTR** ArtCode, **DOUBLE** Qty, **DOUBLE** Value**) ;**

Observations An External File Update message has been received.
This event comes about when an ECR sends a request to an application to update sales data.

Input parameters, those from the SO :

EcrNum = The number of the ECR which sent the request
ArtType = Type of article code :
0 = Numerical
1 = Alphanumerical
ArtCode = String containing the article code to be found
Qty = Quantity sold
Value = Sales value

ProgressCtrl Event

Syntax **VOID ProgressCtrl (LONG** EcrNum, **LONG** Count, **LONG** Total, **BSTR** vString, **LONG** *Abort**);**

Observations The SO sends this when carrying out a batch command:
i.e. Normally a “progress” event is sent every time a new line of the source line is carried out.

Count = If > 0, it automatically shows the number of the source line.
If Count > 100000, this means that we are carrying out an EM_LINK command within a command file.
In this case, Count-100000 = #line in the EM_LINK data file
If = 0, it has a different meaning: shows “internal” message

Total = progress percentage (from 0 to 100)

StatusChange Event

Syntax

Observations This event (only available from version 1.4 on) if there is a change in the state of the driver and/or connected ECRs.

NumEcr = The number of the ECR which has undergone a change
Code = Shows the nature of the change :
Currently the following changes are possible :
0 = one of the ECRs has been suspended/reactivated
Status = Shows a value for the status.
If the event regards an ECR, (Code=0) the value will show
the following bits(weights) :

weight 1 (bit 0) : 0 = ECR not present 1 = ECR present
weight 2 (bit 1) : 0 = ECR operational 1 = ECR suspended (not responding)
weight 4 (bit 2) : U.F.
weight 8 (bit 3) : U.F.
weight 16 (bit 4) : U.F.
weight 32 (bit 5) : U.F.
weight 64 (bit 6) : U.F.
weight 128 (bit 7) : U.F.
Info = Text String containing any information regarding the change which has occurred

In this example, the StatusChange event is used to show the current state of the ECRs connected, based on the colour which appears in the text box txtNEtChange.

18.3 Methods

AboutBox Method

Syntax **VOID AboutBox ();**

Observations This method shows information about CO.

Close Method

Syntax **LONG Close ();**

Observations Called up to release the device and its resources.

Return One of the following values is returned by the method and put in the **ResultCode** property:

Value	**Meaning**
WEC_SUCCESS	The device has been disabled or switched off.
Other Values	See **ResultCode**.

See also **Open** Method

DirectIO Method

Syntax **LONG DirectIO (LONG, LONG*, BSTR*);**

Observations <u>Functions not to be used</u>

EcrStatus Method

Syntax **LONG EcrStatus (LONG** ecrnum**);**

Observations This method shows the state of the ECR, i.e. if it is in general idle, transaction in progress, fatal error etc..

EcrStatusEx Method

Syntax **LONG EcrStatusEx (LONG** ecrnum, **BSTR*** state**);**

Observations This method is like EcrStatus, but with extra information (extended state)

EcrCmd Method

Syntax **LONG EcrCmd (BSTR** command, **BSTR*** response**);**

Observations This method is used to send the SO a command to be carried out. It has two parameters:

The first is a string with the command to be carried out in WinEcrCom format.
The second is a string where WinEcrCom passes on a string which describes the outcome of the operation. The EcrWinCmd returns a "long" which, if different from 0, indicates the code for the error that has occurred. The EcrCmd method comes straight back if the "asynchronous" mode has been activated. In this case the command is carried out in background by the SO which will send a CommandComplete event when it has finished carrying out the command. If in "synchronous" mode, and the EcrCmd comes back as 0, all is well, otherwise an error message comes up.

EcrConfStr Method

Syntax **BSTR EcrConfStr (LONG** value**);**

Observations This method obtains and shows the registration number and version of the ECR currently selected, i.e., knot 1, which is the pre-selected default one upon opening. It has a parameter which allows you to choose which data to find. If the EcrConfVal method comes back with an empty string, it means that it was not possible to read the current ECR configuration. A few obtainable values are:

EcrConfStr(0) = Current driver string status
EcrConfStr(1) = Serial number of selected ECR
EcrConfStr(2) = String of selected ECR model
EcrConfStr(3) = Firmware version string of selected ECR model

EcrConfVal Method

Syntax **LONG EcrConfVal (LONG** value**);**

Observations The EcrConfVal method takes the numerical values from the configuration of the selected ERC. EcrConfVal too has a parameter which specifies which configuration data has to be read. If the EcrConfVal returns the value –1, this means that it was not possible to read the configuration of the ECR currently selected. A few of the values available are:

EcrConfVal(0) = DLL translator version
EcrConfVal(1) = Version of the protocol DLL
EcrConfVal(10) = Valid configuration Flag. 0=invalid

Open Method

Syntax **LONG Open (BSTR** *param***);**
The *param* parameter specifies the name of the device to be opened.

Observations operates a device for the next I/O.
The device name specifies which of the various devices supported by this Control Object is to be used. The *param* must exist in the register of the system for this class of device. The relationship between the name of the device and the physical device is determined by the entrance to the operating system registry. These entrances are maintained by a setup or a configuration utility.

When the **Open** method is successful, it selects the ……..property, the descriptions and the WinEcrCom version number. Additional class-specific properties may also be initialized.

Return One of the following values will be returned by the method.

Value	Meaning
WEC_SUCCESS	Successfully opened.
WEC_E_ILLEGAL	The Control is already opened.
WEC_E_NOEXIST	The specified parameter has not been found.
	WEC_E_NOSERVICEMay not be able to connect to corresponding Service Object..
Other values	see **ResultCode**.

Note
The value of the **Result Code** property after using the **Open** method might not be the same as the value returned by the **Open** method in the following two cases:
1. Control was not on and the **Open** method failed: The **Result Code** property will continue to return WEC_E_CLOSED.
2. Control was already on: the **Open method** will return WEC_E_ELLEGAL, but the **Result Code** property may continue to return the value maintained from before the **Open** method
See also **Close** Method