A6 CCTALK MODULAR

INDICE

1. CCTALK® SERIAL COMMUNICATION PROTOCOL

1.1 Introduction

The communication protocol that has been implemented in the Validator is compatible with **cctalk®**. The specification used for the implementation of this communication protocol is the following: **cctalk® Serial Communication Protocol,** Generic Specification, **Issue 4.3.**

General characteristics of working in serial mode:

- The validator will have a default address assigned in the factory, but this can be changed with cctalk commands ("Address change" "Address poll").
- The validator can work with a total of 16 different types of coins.
- It has 16 inhibit bits, one for each type of coin (command "Modify inhibit status").
- After the switching on the validator or after a reset all the coins will be inhibited.
- The exit code of the coins that are being accepted by the validator can be transmitted to the machine in reply to the command "Read buffered credit or error codes". It is advised that the machine poll the validator with this command at intervals from 100 ms. to 900 ms.. If the time is above 1 second without the machine polling, the validator inhibits the acceptance of all coins until it receives a new "Read buffered credit or error

codes" command. This way of working avoids the validator admitting coins when the machine is not working correctly.

- The Validator has one zone with 64 bytes of data available to the user into the Flash memory. This zone is read and written by the means of the commands "Read data block" and "Write data block".
- Following the CCTALK specifications, Appendix 6, the CCTALK interface identification can be one of the following:

cctalk b96.p0.v12.a5.d0.c5.m0.x8.e0.i3.r4: conector de tipo 5

cctalk b96.p0.v12.a5.d0.c7.m0.x8.e0.i3.r4: conector de tipo 7

The Validator has both types of connectors electrically joined in parallel. The power and communications can be connected indifferently in anyone.

2. PHYSICAL LEVEL

2.1 VOLTAGE LEVELS AND TRANSMISSION SPEEDS

The data is established using TTL levels: 5 volts is a '1' logic and 0 volts is a '0' logic, being the level of rest of 5 volts. The validator has a "pull-up" resistor at 5 Vdc.

Mark (idle): +5Vdc.

Space (activates): 0Vdc

The communication is asynchronous and half-duplex, that is, it cannot simultaneously transmit more than one element on the bus.

The 'timing' of the communication meets the characteristics of the RS232 industrial standard.

RS232 communication has various parameters that in this application are configured as follows:

9600 bauds, 1 start bit, 8 data bits, without parity, 1 stop bit

2.2 CONNECTOR

The validator has a type 5 connector according to the cctalk specification: a 10 pin type "AWP Industry-standard interface". The pin-out is as follows:

PIN	NAME	DESCRIPTION
1	/DATA	Data line
2	N.C.	Not connected
3	N.C.	Not connected
4	N.C.	Not connected. It can be connected to ground by a bridge.
5	N.C.	Not connected
6	N.C.	Not connected
7	VIN	Power supply of the validator
8	GND	Ground, internally connected to 0 V
9	N.C.	Not connected

10	N.C.	Not connected. It can be connected to ground by a bridge.

Table ¡Error!Argumento de modificador desconocido.**: 10 pins CCTALK connector¡**

Type 7 connector, following the CCTALK specification: 4 ways "standard interface, JST connector". The description is as follows:

PIN	**NOMBRE**	**DESCRIPCIÓN**
1	VIN	Power supply of the validator
2	N.C.	Not connected
3	GND	Ground, internally connected to 0 V
4	/DATA	Data line

Tabla ¡Error!Argumento de modificador desconocido.**: 4 pins Conector CCTALK**

3. LOGIC LEVEL

3.1 MESSAGE STRUCTURE

The format of the messages is the following:

[Destination address]

[Nº of data bytes]

[Origin address]

[Header]

[Data 1]

[Data 2]

...

[Data N]

[Checksum]

Each sequence of communication is made up of two strings. The first corresponds to the command sent by the Machine to the Validator, *and* the second is the reply sent by the Validator to the Machine. Both strings have the format indicated previously.

3.2 DESTINATION ADDRESS

The 'Destination address' byte indicates the node on the bus (slave) Where the message is directed. The range of addresses goes from 0 to 255 (of those 254 correspond to Validator addresses as is explained further on).

0: Used in messages that affect all the Validators simultaneously, (Broadcast messages).

1: Machine address*.* When the message is directed from the Validator to the Machine*.*

2 to 255: addresses of the *Validators* in multi-slave communication, configurable using software with MDCES commands. The default address of the Validator is 2 although it can be modified using commands.

3.3 NUMBER OF DATA BYTES

This byte indicates the number of data bytes in the message and not the number total of bytes of the message. If it is '0', it means that the message does not have any data and in this case the number total number of bytes of the message will be of 5 bytes (The minimum permitted).

The values 253 to 255 are not permitted in this field and would be considered as the value 252.

The A6 Validator has the capacity to receive up to a maximum of 50 bytes in each string. In reception, it can receive and calculate the checksum of a string up to 251 bytes, although it can only process the first 46 (see command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**).

3.4 ORIGIN ADDRESS

The byte ‘Origin address’ indicates the node on the bus that sends the message. The range of addresses goes from 1 to the 255 (of those 254 correspond to possible addresses of the *Validator*).

1: address of the *Machine*. When the message is sent by the *Machine.*

2: default address of the *Validator on leaving* the factory, in communication with only one slave. When the message is sent by the *Validator.*

2 to 255: addresses of the *Validator* in multi-slave communication, configurable using software with MDCES commands.

3.5 HEADER

The range of header bytes goes from 0 to 255. The “Header” will never have the value ‘0’ in the case of messages sent by the Machine.

The Validator can return the following values in the header byte:

0: ACK, the Validator has carried out the command correctly

5: NACK, an error has occurred in the processing of the command or the command has not been carried out due to the present validator configuration.

6: BUSY, the Validator is busy doing an operation and cannot attend the command.

3.6 DATA

The range of values that each data byte can take is from 0 a 255.

The meaning of the data depends on the command to execute.

3.7 CHECKSUM

The “Checksum” is the value that makes the 8 lower bits of the sum of all the bytes in the message, including the checksum, give result ‘0’.

For example, the message [01] [00] [02] [00] will be followed by the checksum [253] because:

1 + 0 + 2 + 0 + 253 = 256 = 0.

3.8 TEMPORARY REQUIREMENTS

Maximum time between bytes:

The maximum time between two bytes in the same message is 50 ms. If it is exceeded the communication programme will reset the communication variables and will prepare for the reception of a new message.

Maximum time between command and reply:

The maximum reply time to a command depends on the time that it takes the *Validator* to process the command. This time is, by default, 50 ms, if another value has not been specified in the definition of the command.

Minimum time from the power on until the sending of the first command:

The minimum time it takes from power on to the *Validator* until the first command is sent is **250 ms.**

3.9 ERROR MANAGEMENT

If a Validator receives an incomplete message (reception timeout) or with an incorrect checksum, it will not carry out another action and it will reset the communication variables and prepare to receive a new message. The Machine, on not receiving a reply to any message sent, can choose to resend the same message.

On the other hand, if the Machine receives an incomplete message (reception timeout) or with an incorrect checksum, it can choose to resend the same message.

In any case, when there is an error in the reception of a message, there is no defined NACK message, which simplifies the implementation of multi-slave protocol *and* reduces collisions.

If a Validator receives a command that it is not ready to execute, it responds with a NACK message.

4. START UP PROCEDURE

In the following paragraphs the sequence of commands necessary for the Validator to be able to accept coins after a reset or power-up is indicated. The sequence indicated is the minimum necessary. Another sequence can be used: with a different order of commands and/or more complex.

It is assumed that:

- There is a Validator programmed with the coins to use, powered at the recommended voltage (12 Vdc to 24 Vdc.)
- The CCTALK address of the Validator is known (default: 2)
- The poll time required is known

Sequence of commands:

- ✓ Command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**. This command is only necessary if a PIN protection is used.
- ✓ Command **¡Error!No se encuentra el origen de la referencia.**, see page **¡Error!Marcador no definido.**: if the reply is not OK the Validator will not be able to accept coins. Optionally the command **¡Error!No se encuentra el origen de la referencia.** can be used, page **¡Error!Marcador no definido.**.
- ✓ Command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**: Activate the coins that should be admitted.
- ✓ Command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**. Retransmit this command with adequate frequency (one command in less than 1 second) so that the Validator can accept coins and the machine can read the coin data.

5. LIST OF COMMANDS

All the commands implemented comply with **cctalk®** standards. Among the specific cctalk commands, the first group is general, that is, commands valid for any type of device. The rest of the specific cctalk commands are designed to be used with Validators.

All the commands, except “Address Poll (253)” and “Simple Poll (254)” can be protected by a PIN. If it receives a command and a valid a PIN has not previously been introduced, the Validator will not answer the command. The value a PIN = 0 will deactivate this protection.

¡Error!No se encuentra el origen de la referencia. indicates the codes of the implemented commands.

Table ¡Error!Argumento de modificador desconocido.**: LIST OF COMMANDS**

Header (decimal)	**Command**	**Page**
254	Simple Poll	**¡Error!Marcador no definido.**
253	Address Poll	**¡Error!Marcador no definido.**
252	Address Clash	**¡Error!Marcador no definido.**
251	Address Change	**¡Error!Marcador no definido.**
250	Address Random	**¡Error!Marcador**

		no definido.
249	Request polling priority	**¡Error!Marcador no definido.**
248	Request status	**¡Error!Marcador no definido.**
246	Request manufacturer id	**¡Error!Marcador no definido.**
245	Request equipment category id	**¡Error!Marcador no definido.**
244	Request product code	**¡Error!Marcador**

		no definido.
243	Request database version	**¡Error!Marcador no definido.**
242	Request serial number	**¡Error!Marcador no definido.**
241	Request Software Revision	**¡Error!Marcador no definido.**
240	Test Solenoids	**¡Error!Marcador no definido.**
236	Read opto states	**¡Error!Marcador**

		no definido.
232	Perform self-check	**¡Error!Marcador no definido.**
231	Modify inhibit status	**¡Error!Marcador no definido.**
230	Request inhibit status	**¡Error!Marcador no definido.**
229	Read buffered credit or error codes	**¡Error!Marcador no definido.**
228	Modify master inhibit status	**¡Error!Marcador or**

		no definido.
227	Request master inhibit status	**¡Error!Marcador no definido.**
226	Request insertion counter	**¡Error!Marcador no definido.**
225	Request accept counter	**¡Error!Marcador no definido.**
219	Enter new a PIN number	**¡Error!Marcador no definido.**
218	Enter a PIN number	**¡Error!Marcador or**

		no defin ido.
216	Request data storage availability	**¡Error!Marcador no definido.**
215	Read data block	**¡Error!Marcador no definido.**
214	Write data block	**¡Error!Marcador no definido.**
213	Request option flags	**¡Error!Marcador no definido.**
212	Request coin position	**¡Error!Marcador**

		no definido.
202	Teach mode control	**¡Error!Marcador no definido.**
201	Request teach status	**¡Error!Marcador no definido.**
198	Counters to EEPROM	**¡Error!Marcador no definido.**
197	Calculate ROM checksum	**¡Error!Marcador no definido.**
196	Request creation date	**¡Error!Marcador**

		no definido.
195	Request last modification date	**¡Error!Marcador no definido.**
194	Request reject counter	**¡Error!Marcador no definido.**
193	Request fraud counter	**¡Error!Marcador no definido.**
192	Request build code	**¡Error!Marcador no definido.**
185	Modify coin id	**¡Error!Marcador**

		no defin ido.
184	Request coin id	**¡Error!Marcador no definido.**
181	Modify security setting	**¡Error!Marcador no definido.**
180	Request security setting	**¡Error!Marcador no definido.**
179	Modify bank select	**¡Error!Marcador no definido.**
178	Request bank select	**¡Error!Marcador or**

		no defin ido.
176	Request alarm counter	**¡Error!Marcador no definido.**
170	Request base year	**¡Error!Marcador no definido.**
169	Request address mode	**¡Error!Marcador no definido.**
162	Modify inhibit and override registers	**¡Error!Marcador no definido.**
141	Request firmware upgrade capability	**¡Error!Marcador**

		no definido.
140	Upload firmware	**¡Error!Marcador no definido.**
139	Begin firmware upgrade	**¡Error!Marcador no definido.**
138	Finish firmware upgrade	**¡Error!Marcador no definido.**
135	Set acceptance limit	**¡Error!Marcador no definido.**
99	Begin tables upload	**¡Error!Marcador**

		no definido.
98	Upload tables	**¡Error!Marcador no definido.**
97	Finish tables upload	**¡Error!Marcador no definido.**
96	Request modules information	**¡Error!Marcador no definido.**
4	Request comms revision	**¡Error!Marcador no definido.**
3	Clear comms status variables	**¡Error!Marcador**

		no definido.
2	Request comms status variables	**¡Error!Marcador no definido.**
1	Reset device	**¡Error!Marcador no definido.**

5.1 SIMPLE POLL [254]

Command for checking communication is working correctly and for confirming the presence in the bus of a Validator.

Send:	[Dir] [0] [1] [254] [Chk]
Reply:	[1] [0] [Dir] [0] [Chk] → ACK without data

In the case that no answer is received to the request sent (reception timeout in the Machine) it will be a sign that the corresponding validator is faulty or not connected.

The Validator always responds to this command: command not protected by a PIN.

5.2 ADDRESS POLL [253]

This command is used to request all the slave devices to return their addresses. To do this, it is sent with destination address 0 (Broadcast). To avoid collisions, only the byte of address is returned with a delay proportional to the value of the address.

Send: [0] [0] [1] [253] [2]

Reply: {Delay variable} [Dir]

Where: Dir = address of the corresponding validator

The algorithm to calculate the delay with which it responds is the following:

```
Deactivate port rx
Delay (4*Dir) ms
Send [Dir]
Delay 1200 – (4*Dir) ms
Activate port Rx
```

If the Machine receives all the bytes 1.5 seconds after sending the command, it can determine the quantity and the addresses of the devices connected.

The Validator always responds to this command: command not protected by a PIN

Notes:

a) The Validator will not receive any more commands until 1.2 seconds after having received the command.

b) This command is the only one not protected by a PIN.

5.3 ADDRESS CLASS [252]

This command is used to check if one or more devices share the same address. The difference to the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**, is it sends a concrete address.

To avoid collisions, only the address byte is returned with a random delay in the sending.

Send: [Dir] [0] [1] [252] [Chk]

Reply: {Delay variable} [Dir]

Where: Dir = address of the *corresponding validator*

The algorithm used to calculate the delay with which it responds is the following:

```
R = rand (256)
Deactivate port Rx
Delay (4 x R) ms
Send [Dir]
Delay 1200 – (4 x R) ms
Activate port Rx
```

If the Machine receives all the bytes 1.5s after sending the command, it can determine the quantity of devices connected that share the same address.

Although there exists the possibility that two devices share the same address generate the same random number, the probability that this occurs is very small (1 in 254*256 = 1 in 65.024).

Command protected by a PIN.

Notes:

a) The Validator will not receive any more commands until 1.2 seconds after having received the command.

5.4 ADDRESS CHANGE [251]

This command permits the programming of a new address to the Validator, valid for all the commands that it receives. In case that the new address is 0 or 1, the Validator will take the default address as its new address. The address is stored in the Flash memory of the Validator.

Send: [Dir] [1] [1] [251] [Data 1] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where: Dir = address of the corresponding validator

Data 1 = new address of the corresponding validator

Command protected by a PIN.

Notes:

a) The reply ACK is carried out with the original address [Dir]. The following commands to be transmitted will only be accepted if they are directed at the new address [Data 1].

b) Once this command is used with this option (new address = 0 or 1) and when there is the possibility that the Machine does not know the new default address of the Validator, the Machine should send the command **¡Error!No se encuentra el origen de la referencia.**, see page **¡Error!Marcador no definido.**.

c) If the Validator cannot save the new value of the address in its memory a NACK is returned.

***Example*:**

The Machine programmes the Validator whose address is 3 with the new address 13

Send: [3] [1] [1] [251] [13] [243]

Reply: [1] [0] [3] [0] [252]

After the reply the Validator is programmed with address 13

5.5 ADDRESS RANDOM [250]

This command allows the Validator to be programmed with a random new address. This is last resort for cases when various devices share the same address, after their new addresses are requested. The address is stored in the FLASH memory of the Validator

Send: [Dir] [0] [1] [250] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where: Dir = address of the corresponding validator

Command protected by a PIN.

Notes:

a) The reply ACK is carry out with the original address original [Dir].

b) Once this command is used the Machine should send the command **¡Error!No se encuentra el origen de la referencia.** (see page **¡Error!Marcador no definido.**) to know the new address of the Validator. The Validator does not use the addresses 0 and 1.

c) If the Validator cannot save the new value of the address in its memory NACK is returned.

***Example*:**

The Machine indicates to the Validator whose address is number 3 that it will be programmed with a random value.

Send: [3] [0] [1] [250] [2]

Reply: [1] [0] [3] [0] [252]

After the reply the Validator is programmed with a random address.

5.6 REQUEST POLLING PRIORITY [249]

This command requests the Validator for information about the interval of time recommended to carry out the Request (Polling) and avoid the loss of credit or error code data (see command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**).

Send: [Dir] [0] [1] [249] [Chk]

Reply: [1] [2] [Dir] [0] [Data 1 = 2] [Data 2 = 20] [Chk] -> ACK with 2 data

Where Dir = address of the corresponding validator

Data 1: units of time used = 2 (units of 10 ms)

Data 2: value in units of time = 20 (20 units of 10 ms = 200 ms)

Command protected by a PIN.

5.7 REQUEST STATUS [248]

This command requests the Validator for information about its status: string detector system and/or coin return mechanism activated.

Send: [Dir] [0] [1] [248] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1] [Chk] -> ACK with 1 data

Where Dir = address of the *corresponding validator*

Data 1 = Status shown in **¡Error!No se encuentra el origen de la referencia.**:

Code	Status
0	O.K.
1	Coin return mechanism activated
2	String detector system activated

Table ¡Error!Argumento de modificador desconocido.**: Status codes**

Command protected by a PIN.

Notes:

a) This command will not carry out a complete check of the validator. For this use the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.** is recommended.

b) If the Validator does not have its software correctly loaded, a NACK is returned to this command.

5.8 REQUEST MANUFACTURER ID [246]

With this command, the corresponding validator returns the manufacturer identification of the device to the Machine: “AZK”

Send: [Dir] [0] [1] [246] [Chk]

Reply: [1] [6] [Dir] [0] [Data 1] ... [Data n] [Chk] -> ACK with n data

Where: Data 1 - Data n = string of ASCII characters

Dir = address of the corresponding validator

The default value returned consists of 3 bytes:

Data 1 = ‘A’

Data 2 = ’Z’

Data 3 = ’K’

Two strings can be returned: the default value – 'AZK' and a string whose value is programmed in the Factory (maximum 10 characters)

Command protected by a PIN.

Notes:

a) If the Exit module does not have their tables correctly programmed the default value ("AZK") is returned. If it is correctly programmed the value programmed in Factory is returned.

5.9 REQUEST EQUIPMENT CATEGORY ID [245]

This command permits the reception of a string of characters from the Validator that identifies the type of device it is: 'Coin Validator'

Send: [Dir] [0] [1] [245] [Chk]

Reply: [1] [13] [Dir] [0] ['C'] ['o'] ['i'] ['n'] [' '] ['A'] ['c'] ['c'] ['e'] ['p'] ['t'] ['o'] ['r'] [Chk] -> ACK with 13 data

Where: Dir = address of the *corresponding validator*

Command protected by a PIN.

5.10 REQUEST PRODUCT CODE [244]

With this command, the Validator returns the code of product of the device to the Machine. The complete identification of the product can be determined using the command REQUEST PRODUCT CODE followed by the command **¡Error!No se encuentra el origen de la referencia.** (page **¡Error!Marcador no definido.**).

Send: [Dir] [0] [1] [244] [Chk]

Reply: [1] [n] [Dir] [0] [Data 1] ... [Data n] [Chk] -> ACK with n data

Where: Dir = address of the *corresponding validator*

Data 1 - Data n = ASCII characters

n = Quantity of data to send

Two strings are returned: the default value (‘A6–CCTALK’: 9 characters) and a string whose value is programmed in the Factory or with the user tools (maximum 20 characters)

Command protected by a PIN.

Notes:

a) If the Exit module does not have its tables correctly programmed the default value (“Z6–BACTA/CCTALK”) is returned. If it is correctly programmed the value programmed in the Factory or with the user tools is returned.

5.11 REQUEST DATABASE VERSION [243]

This command requests the Validator for information about the data base number that should used for the tele-programming, between 1 and 255. This number corresponds to the table version stored in the Sensor module fitted in the Validator.

Send: [Dir] [0] [1] [243] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1] [Chk] -> ACK with 1 data

Where Dir = address of the *corresponding validator*

Data 1 = 1 a 255 : table version of the Sensor module

Command protected by a PIN.

a) A NACK is returned if:

- The Sensor module is not programmed or it is incorrectly programmed.

- The Sensor module is not connected or it does not work correctly. In this case the reply time of the NACK is 1.2 seconds.
- The Validator (Exit module) does not have the software correctly loaded.

5.12 REQUEST SERIAL NUMBER [242]

In reply to this command, the serial number of the Sensor module is returned. This number is required to generate the corresponding data for tele-programming this Sensor module in the Factory. The serial number consists of 4 bytes:

Send: [Dir] [0] [1] [242 [Chk]

Reply: [1] [3] [Dir] [0] [LSB] [2SB] [3SB] [4SB] [Chk] -> ACK with 4 data

Where Dir = address of the Validator.

LSB, 2SB, 3SB, MSB: bytes of the serial number (4 bytes)

Command protected by a PIN.

a) A NACK is returned if:

- The Sensor module is not calibrated or it is incorrectly calibrated.
- The Sensor module is not connected or it does not work correctly. In this case the reply time of the NACK is of 1.2 seconds.
- The Validator (Exit module) does not have the software correctly loaded.

5.13 REQUEST SOFTWARE REVISION [241]

With this command, the corresponding validator returns the present software version of the device to the machine.

Send: [Dir] [0] [1] [241] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1] ... [Data n] [Chk] -> ACK with n data

Where: Dir = address of the corresponding validator

Data 1 - Data n = ASCII characters

n = Quantity of data to send

The data to send consists of:

- Product code: same string that the command **¡Error!No se encuentra el origen de la referencia.** returns, page **¡Error!Marcador no definido.**.
- String 'Vxx.y' indicating the software version of the Exit module. 'xx' corresponds to the version. '.' is a separator. 'y' corresponds to the subversion.

The string with the product code has the same restrictions as the command **¡Error!No se encuentra el origen de la referencia.**.

Command protected by a PIN.

Example:

The Machine requests *Validator* number 2 for its software version, which in this example is 'Z6-BACTA/CCTALK V1.0'.

Send: [2] [0] [1] [241] [12]

Reply: [1] [14] [2] [0] ['A'] ['6'] ['-'] ['C'] ['C'] ['T'] ['A'] ['L'] ['K'] [' '] ['V'] ['1'] ['.'] ['0'] [148]

5.14 TEST SOLENOIDS [240]

Using this command the different solenoids of the validator are activated for a determined time: The acceptance gate and 3 sorter solenoids. The bit mask that is sent indicates which solenoids are activated.

Send: [Dir] [1] [1] [240] [Data 1] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the corresponding *Validator*

[Data 1] = Bit mask of solenoid activation

Bit 0 – Acceptance gate solenoid

The acceptance gate solenoid is activated for 500 ms. The ACK reply of the *Validator* is produced after the end of the command.

Bit 0 of [Data 1] must be at “1”.

Command protected by a PIN.

Notes:

a) If the Exit module does not have its software correctly loaded a NACK is returned.

b) If there is a coin inside or the Validator is activating the exits or the solenoids at the moment of reception of the command a NACK is returned.

c) There should be a visual or sound check that the corresponding mechanism activates and deactivates.

Example:

The Machine requests Validator 2 to carry out a test of the acceptance gate solenoid and solenoid 1 of the sorter.

Send: [2] [1] [1] [240] [3] [9]

Reply: [1] [0] [2] [0] [253] after 500 ms.

Note: Check visually that for 500 ms the path of acceptance opens and it moves the sorter.

5.15 READ OPTO STATES [236]

Using this command, the Validator retrns a byte which indicates the value of the optic sensors in the validator: Photo transistors detecting Coin Exit, String detector and Double Entry.

Send: [Dir] [0] [1] [236] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1] [Chk] -> ACK with a data

Where: Dir = address of the corresponding validator

Data 1 = Values

Bit 0: Not used

Bit 1: Not used

Bit 2: Exit phototransistor

Bit 3: String detector mechanism

Bits 4, 5, 6 and 7: not used = 0

If the optic beam is blocked the value of the corresponding bit is 1 and is returned 0 if the beam is not blocked.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded.

b) The String detector mechanism is normally “closed”. Bit 3 at “0” indicates the normal value. If the value returned is 0 it means that all the optics beams are in the correct state.

Example:

The Machine requests *Validator* number 2 for the value of the Optos.

Send: [2] [0] [1] [236] [17]

Reply: [1] [1] [2] [0] [12] [249]

Note: in this reply (12 = 0x0C) we deduce that the string detector mechanism is opened and that the coin exit optic beam is blocked.

5.16 PERFORM SELF-CHECK [232]

Using this command the Validator carries out an auto check or hardware test. The validator will return information on the diagnostics carried out using error codes of 1 or 2 bytes.

Send: [Dir] [0] [1] [232] [Chk]

Reply: [1] [1/2] [Dir] [0] [Data 1] / [Data 2] [Chk] -> ACK with 1 or 2 data

Where Dir = address of the *corresponding validator*

Data 1 = Error code

Data 2 = Complementary error code

The different error codes are presented in **¡Error!No se encuentra el origen de la referencia.**.

Code	**Type of Error [Data 1]**	**Complementary Information [Data 2]**
0	OK (no fault has been detected)	-
2	Fault in electromagnetic sensors	Bit 0: sensor 1 Bit 1: sensor 2 Bit 3: sensor 3
3	Fault in credit sensor: optic beam of coin exit sensor covered broken	-

4	Fault in sound sensor or piezoelectric of the Sensor module	-
6	Fault in diameter sensor	-
20	Error in the string detector mechanism (it is open)	-
28	Sensor module does not respond: not connected or not responding.	-
33	Fault in power supply voltage of the Sensor module	-
34	Fault in temperature sensor of the Sensor module.	-
255	Hardware test not valid: Validator is measuring a coin in the Sensor module	-

Table ¡Error!Argumento de modificador desconocido.**: Hardware Test – Error Codes**

If the code 255 is returned (Code of a non specific fault), it means that is has carried out a hardware test with a coin in the interior of the Sensor module. It is necessary to carry out various retries before putting the validator out of order.

When there is the possibility of putting the validator out of order deliberately, it seems necessary to establish a reset mechanism when the validator responds to a new test without errors. It is also recommended to try a new test before communicating the fault of the validator to base.

In the case of various simultaneous errors, the process of the hardware test finalises on detecting the first error without checking the rest.

The reply to this hardware test is received when the test is completed.

Typical time of reply: 25 ms.

1.2 seconds if the Sensor module is not connected or it does not work correctly.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded.
b) The reply is one byte, with the exception of error in the electromagnetic sensors and in the double entry sensor, where two bytes are returned.

Example:

The Machine requests to Validator 2 to carry out a Hardware Test:

Send: [2] [0] [1] [232] [21]

Reply: [1] [2] [2] [0] [2] [3] [246]

The *Validator* informs that it has detected an error in electromagnetic sensor number 3: (reply = 3: 0000.0100B)

After correcting the error

Send: [2] [0] [1] [232] [21]

Reply: [1] [1] [2] [0] [0] [252] -> Validator OK

5.17 MODIFY INHIBIT STATUS [231]

Using this command the 16 inhibit bits of the Validator configured. Each coin table will be associated with an inhibit bit, so that the activated coins will be those whose inhibit bit is at 1. Initially and after a reset all the bits are at 0, so that all the coins will be inhibited.

Send: [Dir] [2] [2] [231] [Data 1] [Data 2] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the corresponding *Validator*

[Data 1] = Inhibit byte 1 (LSB), coins 1 to 8

[Data 2] = Inhibit byte 2 (MSB), coins 9 to 16

Each coin is indicated by a bit. Seeing Data 1 and Data 2 as a whole number without sign:

- bit 0 (in Data 1): coin 1
- bit 1 (in Data 1): coin 2
- ...
- bit 7 (in Data 1): coin 8
- bit 8 (in Data 2, equivalent to bit 0 of Data 2): coin 9
- bit 9 (in Data 2, equivalent to bit 1 of Data 2): coin 10
- ...
- bit 15 (in Data 2, equivalent to bit 7 of Data 2): coin 16

The inhibit data is saved in RAM.

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded

- The Exit module is programmed to work in Mixed mode (parallel and serial) or only in "parallel" mode.

b) After a reset all the coins are deactivated (the inhibit bits are deleted). As a consequence it is necessary to activate the coins using this command so that the Validator can accept coins. We can check the value of the inhibit bits using the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Example:

The Machine configures Validator number 2 to activate tables 1, 2, 3, 6, 8, 10 and 12:

- Data 1 = 167 (1010.0111B)
- Data 2 = 10 (0000.1010B)

Send: [2] [2] [1] [231] [167] [10] [99]

Reply: [1] [0] [2] [0] [253]

5.18 REQUEST INHIBIT STATUS [230]

This command requests the Validator for information about the configuration of the inhibit bits of the coins. This command is complementary to **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Send: [Dir] [0] [1] [230] [Chk]

Reply: [1] [2] [Dir] [0] [Data 1] [Data 2] [Chk] -> ACK with 2 data

Where Dir = address of the corresponding validator

[Data 1] = Inhibit Byte 1 (LSB: coins 1 to 9)

[Data 2] = Inhibit Byte 2 (MSB: coins 10 to 16)

The coins that are activated have the bit at “1”. Data 1 and Data 2 have the same meaning as in the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

b) After a reset all the coins are deactivated (the inhibit bits are deleted). As a consequence is necessary activate the coins using this command so that the Validator can accept coins. We can modify the value of the inhibit bits using the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Example:

The Machine requests Validator number 2 for information about the inhibit bits.

Send: [2] [0] [1] [230] [23]

Reply: [1] [2] [2] [0] [0] [0] [251]

From this reply we deduce that the Validator has all the coins inhibited: a normal situation after a reset.

5.19 READ BUFFERED CREDIT OR ERROR CODES [229]

This command requests the Validator for information about the last events occurred while validating coins and these are saved in a 10-bytes buffer, storing the 5 last events. This allows the Machine to carry out the request at a speed faster than the introduction of coins and not lose any information of credits or events.

Send: [Dir] [0] [1] [229] [Chk]

Reply: [1] [11] [Dir] [0] [Data 1] [Data 2] ... [Data 11] [Chk] -> ACK with 11 data

Where Dir = address of the *corresponding validator*

Data 1 = event counter: 0 – on or reset, 1 to 255 – counter.

Data 2 = result 1A: 0 – there is an error, 1 to 16 – credit (output code of the coin).

Data 3 = result 1B: 1 to 255 – error code, 1 to 5 – path of sorter......

Data 10 =result 5A

Data 11 =result 5B

A new event makes the data rotate in the buffer, and the last event is lost.

The event counter indicates to the Machine any new event that happens in the validator, and for each request it should compare the counter with the last known value.

The event counter is incremented each time that a new credit or error is added to the buffer. When the counter reaches 255, the following event makes the counter take the value 1. The only way the event counter is at 0 is that there has been a power or reset. It is a good way of informing the Machine of a fault in the power supply.

Each event is represented by two bytes (x = 1, 2, 3, 4, and 5):

- result(x)A: indicates that a error has occurred (value = 0)

  code of the coin that has been introduced (value = 1 to 16)
- result(x)B: error code (if result(x)A = 0). See **¡Error!No se encuentra el origen de la referencia.**.

= 0 if there is not an error.

In **¡Error!No se encuentra el origen de la referencia.** the error codes that give information on the possible causes of the rejection of the coin are presented.

Table ¡Error!Argumento de modificador desconocido.**: Command Read Buffered Credit: Error codes**

Code	Error
0	Empty event (no error)
1	Coin Rejected (Sensor module is not correctly programmed or the coin corresponds to a wider acceptance band than the expected)
2	Coin inhibited
5	Timeout error on validation of coin (coin rejected)
6	Timeout in exit detection
8	Error of two passing coins too closely together
13	Sensor module does not work correctly
14	Coin jam in exit detector
20	The String detector has been activated
23	Coin too fast through the exit detector
25	Coin Rejected (not in tables, fraud attempt)
128	Coin 1 inhibited by the inhibitions register
..	...
127+ n	Coin ‘n’ Coin 1 inhibited by the inhibitions register
...	...
143	Coin 16 inhibited by the inhibitions register
254	Refund mechanism activated

255	Error code not specified

No coin is accepted until the command READ BUFFERED CREDIT OR ERROR CODES [229] is received.

Each time the first command is received (229) the validator can accept coins and starts a timer that if **1 second** passes without receiving this command the validator stops accepting coins. This action is carried out so that events of coins are not lost if the Machine cannot consult the Validator with sufficient frequency. The advised consulting time can be obtained using the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**. On receiving command 229 the timer is reset.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

b) After a reset or power-up deactivates the admission of coins until it receives the command READ BUFFERED CREDIT OR ERROR CODES [229].

Example:

The Machine requests Validator number 2 for information on credits or errors.

Send: [2] [0] [1] [229] [24]

Reply: [1] [11] [2] [0] [6] [6] [4] [0] [20] [0] [23] [6] [4] [13] [2] [158]

Note: From this reply we deduce that the validator has detected 6 events and the last five have been: coin 6 accepted through path 4, coin rejected by string detector, coin rejected as not in tables, coin 6 accepted through path 4 and coin 13 accepted through path 2.

5.20 MODIFY MASTER INHIBIT STATUS [228]

The Validator has a “master inhibit” bit, such that if it is deactivated the Validator will not accept any coin. This bit can be activated or deactivated using this command:

Send: [Dir] [1] [2] [228] [Data 1] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the corresponding validator

Only bit 0 of Data1 is used.

Data1.bit0 = 0: Inhibit activated, coins not accepted

Data1.bit0 = 1: Normal working mode.

The “Master Inhibit” bit is saved in RAM and is reset to 1 on reset or power-up.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

5.21 REQUEST MASTER INHIBIT STATUS [227]

Request information from the validator about of the present state of the “Master Inhibit” bit:

Send: [Dir] [0] [2] [227] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1][Chk] -> ACK with 1 data.

Where Dir = address of the corresponding validator

Data1.bit0 = 0: Inhibit activated, coins not accepted

Data1.bit0 = 1: Normal working mode.

This bit is modified using the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

b) This bit will be at 0: coins inhibited, after a reset or power-up.

5.22 REQUEST INSERTION COUNTER [226]

This command requests the Validator for information about the counter of coins inserted:

Send: [Dir] [0] [1] [226] [Chk]

Reply: [1] [3] [Dir] [0] [Data 1] [Data 2] [Data 3] [Chk] -> ACK with 3 data

Where Dir = address of the *corresponding validator*

Data 1	= number of coins inserted (LSB)
Data 2	= number of coins inserted (2SB)
Data 3	= number of coins inserted (MSB)

The validator keeps a counter with the number of coins inserted. This counter is of three bytes and can store up to 16.777.215 coins. It is incremented each time a coin starts to be measured in the Validator.

This counter, together with the coins accepted, rejected and frauds counter is reset to the default 0 on a reset or power-up. However, if the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**, is used the counters are saved in the Flash memory of the Validator and are recuperated after each reset or power-up.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

Example:

The Machine requests Validator number 2 for information about the coins inserted counter.

Send: [2] [0] [1] [226] [27]

Reply: [1] [3] [2] [0] [208] [57] [2] [197]

From this reply we deduce that the validator has detected the insertion of 145.872 coins: LSB = 208, 2SB = 57, MSB = 2.

If now the command **¡Error!No se encuentra el origen de la referencia.** (see page **¡Error!Marcador no definido.**) is used:

Send: [2] [0] [1] [198] [55]

Reply: [1] [0] [2] [0] [253]

The validator has stored the value of the counters in its Flash memory. If now we switch off the validator, in the power-up the value of the counters in the Flash memory in read and the coins inserted counter will have the value that it had when the command COUNTERS TO EEPROM: 145.872 was executed.

5.23 REQUEST ACCEPT COUNTER [225]

This command requests the Validator for information about the coins accepted counter.

Send: [Dir] [0] [1] [225] [Chk]

Reply: [1] [3] [Dir] [0] [Data 1] [Data 2] [Data 3] [Chk] -> ACK with 3 data

Where Dir = address of the *corresponding validator*

Data 1	= number of coins accepted (LSB)
Data 2	= number of coins accepted (2SB)
Data 3	= number of coins accepted (MSB)

The validator keeps a counter with the number of coins accepted. This counter is of three bytes and can store up to 16.777.215 coins. It is incremented each time a coin is accepted by the Validator.

This counter works the same as the coins inserted counter. See the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

5.24 ENTER A NEW PIN NUMBER [219]

The Validator can be protected by a number (a PIN) so that no command can be executed (except the command **¡Error!No se encuentra el origen de la referencia.**, see page **¡Error!Marcador no definido.**) if the correct PIN has not been previously introduced.

The command ENTER A NEW PIN NUMBER allows the modification of the PIN stored in the Validator:

Send:	[Dir] [4] [1] [219] [Data 1] [Data 2] [Data 3] [Data 4] [Chk]	
Reply:	[1] [0] [Dir] [0] [Chk]	-> ACK without data

Where	Dir = address of the corresponding validator
	Data 1, Data 2, Data 3, Data 4: new PIN number

This command is protected by a PIN, so to execute it, it is necessary to have previously introduced the PIN stored in the Validator (see command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

The PIN is stored in the Flash memory of the Validator. The PIN value = 0,0,0,0 deactivates the protection of the PIN.

Notes:

a) A NACK is returned if an error has occurred when programming the PIN in the Flash memory.

5.25 ENTER A PIN NUMBER [218]

The Validator can be protected by a number (a PIN) so that no command can be executed (except the command **¡Error!No se encuentra el origen de la referencia.**, see page **¡Error!Marcador no definido.**) if the correct PIN has not previously been introduced.

The command ENTER A PIN NUMBER allows the introduction of a PIN that is stored in the Flash memory of the Validator to be used to execute commands if the protection by a PIN is activated:

Send: [Dir] [4] [1] [218] [Data 1] [Data 2] [Data 3] [Data 4] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the corresponding validator

Data 1, Data 2, Data 3, Data 4: PIN number

This command always responds ACK, independently of if the PIN introduced is correct or not.

5.26 REQUEST DATA STORAGE AVAILABILITY [216]

The Validator has one zone into its Flash memory used only for data storage from the Machine. Using this command the Machine can be informed about the type of memory and the capability of this zone:

Send: [Dir] [0] [1] [216] [Chk]

Reply: [1] [5] [Dir] [0] [Data 1] [Data 2] [Data 3] [Data 4] [Data 5][Chk] -> ACK with 5 data

Where Dir = address of the corresponding Validator

Data 1 = 2: Type of memory: non-volatile (permanent), with limited use in number of writes.

Data 2 = 2: Number of blocks of read data available.

Dato 3 = 32: Number of bytes per block of read data..

Dato 4 = 2: Number of blocks of write data available.

Dato 5 = 32: Number of bytes per block of write data.

The user zone is 64 bytes long. It is contiguous and it is divided in 2 blocks of 32 bytes each. Both blocks are read/write enabled and they are identified as 0 and 1.

The user zone is part of the Flash memory of the microcontroller of the Exits Module of the Validator. This memory lasts 10.000 writes (typical). The stored data are not modified if the Validator is powered-off and they are available after the power-on. The contents of this zone are not modified if the Sensor Module is changed or modified, neither when the Validator is teleprogrammed nor when the firmware is updated.

The contents of this zone can be modified (erased to 0) if the Exits Module is damaged and it implies to replace the microcontroller or reprogram the firmware in factory.

In the paragraphs **¡Error!No se encuentra el origen de la referencia. ¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.** and **¡Error!No se encuentra el origen de la referencia. ¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.** the details of use of the read and write in this zone are written.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

5.27 READ DATA BLOCK [215]

Using this command the Machine can read the stored data in the user zone:

Send: [Dir] [N] [1] [215][Data 1][Data 2 (optional)] [Chk]

Reply: [1] [M] [Dir] [0] [Data 3] … [Data (M+2)][Chk] -> ACK with 'M' data

Where Dir = address of the corresponding Validator

N = 1: if [Data 2] is not transmitted (CCTALK standard)

2: if [Data 2] is transmitted.

Data 1 = 0, 1: block number to read

Data 2 (optional) = number of bytes to be read from the block indicated in Data 1.

M: Number of read bytes. If the optional argument is not present: [Data 2], then M = 32 bytes. If it is present then M = [Data 2].

The optional argument: [Data 2] can have the following values:

- 1 .. 64 if the block 0 is read: [Data 1] = 0

- 1 .. 32 if the block 1 is read: [Data 1] = 1

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded

- It is requested to read data out of the user zone or the number of bytes to be read is 0.

5.28 WRITE DATA BLOCK [214]

Using this command the Machine can write data into the user zone:

Send: [Dir] [N] [1] [214][Data 1][Data 2] … [Data M][Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the corresponding Validator

Data 1 = 0, 1: block number to write

Data 2 .. Data M = data to write.

The number of data to write is:

- 1 .. 64 if the write is in the block 0 [Data 1] = 0. If more than 32 bytes are written, the rest of data are written into the block 1.
- 1 .. 32 if the write is in the block 1: [Data 1] = 1.

Because the microcontroller writes into the Flash memory in blocks of 64 bytes long and with the aim to reduce the writes number, it is recommended to write 64 bytes at block 0.

A possible method of writing can be the following one

a) Read the 64 data bytes: command "Read data block" (page **¡Error!Marcador no definido.**).
   [Dir] [2] [1] [215][block = 0][num. of Bytes = 64] [Chk]

b) Modify the received data with the new data.

c) Write the 64 data bytes: command "Write data block"
   [Dir] [65] [1] [214][block = 0][Data 1] … [Data 64][Chk]

The Validator checks the write verifying that the written data are the same as the received data. The Validator transmits ACK if there is no errors.

Command protected by a PIN. The response time is 5 mseg. approximately

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded
- It is requested to write data out of the user zone or the number of bytes to write is 0.
- If an error has occurred while the Flash memory is programmed.

5.29 REQUEST OPTION FLAGS [213]

Using this command the Validator informs the Machine how to send the coin values:

Send: [Dir] [0] [1] [213] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1] [Chk] -> ACK with a data

Where Dir = address of the corresponding validator

Data 1 = 0: the coin is indicated as “position of coin”

The codes of transmitted coins correspond to the position that they occupy in the memory of the Validator. Each code corresponds with the position indicated in the inhibit bits (see commands **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.** and **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**).

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded.

5.30 REQUEST COIN POSITION [212]

Using this command the Validator informs the Machine what the inhibit bit corresponding to the coin code is:

Send: [Dir] [1] [1] [212] [Data 1] [Chk]

Reply: [1] [2] [Dir] [0] [Data 2] [Data 3] [Chk] -> ACK with two data

Where	Dir = address of the corresponding validator
	Data 1: coin code
	Data 2: LSB of the inhibit word
	Data 3: MSB of the inhibit word

The value returned (2 bytes: word) has the bit corresponding to the coin code requested activated to “1”. The values correspond to the following commands:

- Coin code: code returned in the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.
- Inhibit word: value of the coin inhibits – coins to admit or reject, used in the commands **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.** and **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if:
   - The Exit module does not have its software correctly loaded.
   - The coin code is greater than 16 (erroneous argument)

5.31 TEACH MODE CONTROL [202]

The user can programme two coins or tokens in the field. The Machine indicates to the Validator that it should start Auto programming of coins mode:

Send: [Dir] [1] [1] [202] [Data 1] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the corresponding validator

Data 1 = Indicates the position of the table that is to be programmed

Once the Validator has responded ACK a coin can be introduced so that the Validator can recognise it. It has following limitations:

- The codes of the auto programmable coins must be preprogrammed at factory.
- The Validator has to be previously programmed in the Factory to permit the auto programming of both coins.
- 25 of the same coin have to be introduced. The auto programming finishes automatically on introducing the 25th coin. The coins must not have defects such as dents of deformations as they can provoke elevated dispersion of the measured values.
- All introduced coins will be accepted by the Validator.

The state of the auto programming of coins can be consulted using the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded.
- The Validator is not programmed in Factory to accept the auto programming of the coin requested.
- There has been an error in programming the data in the Flash memory of the Sensor module or of the Exit module.

b) BUSY is returned (6) if the Validator is busy measuring or accepting a coin at the moment of receiving the command.

5.32 REQUEST TEACH STATUS [201]

The Machine requests the Validator for information about the auto programming process.

The process of auto programming can be finalised by the Validator or the Machine can also abort it at any given moment.

Send: [Dir] [1] [1] [201] [Data 1] [Chk]

Reply: [1] [2] [Dir] [0] [Data 2] [Data 3] [Chk] -> ACK with 2 data

Where Dir = address of the *corresponding validator*

Data 1 = 0: Indicates if the machine is requesting information.

= 1: Indicates that the machine is aborting the process of auto programming.

Data 2 = Number of coins introduced.

Data 3 = code of state as explained in **¡Error!No se encuentra el origen de la referencia.**:

Table ¡Error!Argumento de modificador desconocido.**: Auto programming state**

Code	Error
252	Auto programming cancelled
253	Error in the Auto programming
254	Auto programming in progress
255	Auto programming completed

As the possibility of the machine finishing the Auto programming command does not exist, this is finalised on introducing 25 coins. On introducing the last coin it will send the message “Auto programming completed” to the machine.

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded.
- An error has occurred in programming the data in the Flash memory of the Sensor module or of the Exit module.

b) BUSY is returned (6) if the Validator is busy measuring or accepting a coin at the moment of receiving the command.

5.33 COUNTERS TO EEPROM [198]

The Machine requests the Validator to store in memory no volatile, in this case the Flash memory of the Validator, the counters for:

- Coins inserted
- Coins accepted
- Coins rejected
- Non valid coins (frauds)

Send:	[Dir] [0] [1] [198] [Chk]
Reply:	[1] [0] [Dir] [0] [Chk] -> ACK without data

At the moment of receiving the command it writes the present value of the counters in memory. Said counters are read from the Flash memory immediately after a reset or power-up. Initially they are at 0.

Command protected by a PIN.

Notes:

a) A NACK is returned if

- The Exit module does not have its software correctly loaded.
- There has been an error in programming the data in the Flash memory of the Sensor module or of the Exit module.

5.34 CALCULATE ROM CHECKSUM [197]

With this command the Validator returns information about the checksum of the ROM memory of the Validator Exit module to the Machine, in 4 bytes:

Send: [Dir] [0] [1] [197] [Chk]

Reply: [1] [4] [Dir] [0] [Data 1] [Data 2] [Data 3] [Data 4] [Chk] -> ACK with 4 data.

Where Dir = address of the *corresponding validator*

Data 1- Data 4 = Cheksum1 (LSB) – Cheksum4 (MSB)

The Validator or more precisely, the Exit module carries out a sum in an “unsigned long” of the ROM memory content. The result of the sum is returned in the reply to the command.

Maximum time of reply: 150 ms.

Command protected by a PIN.

Notes:

a) The checksum only includes the programme area of the Exit module. The tables and data programming are not included. In consequence, any modification of the tables in the Validator or of the data does not modify the checksum calculated.

5.35 REQUEST CREATION DATE [196]

This command requests the Validator for information about the creation date of the programming of the coin tables of the Validator:

Send: [Dir] [0] [1] [196] [Chk]

Reply: [1] [2] [Dir] [0] [Data 1] [Data 2] [Chk] -> ACK with 2 data

Where Dir = address of the *corresponding validator*

Data 1 = LSB of the date.

Data 2= MSB of the date.

The format of the date is indicated in **¡Error!No se encuentra el origen de la referencia.**:

Table ¡Error!Argumento de modificador desconocido.**: Date format**

FORMAT of the DATE			
Bit 15 Bit 14	Bit 13 Bit 9	Bit 8 Bit 5	Bit 4 Bit 0
Reserved	Year	Month	Date
	Relative to the base year	1 to 12	1 to 31

The base year is executed in the command **¡Error!No se encuentra el origen de la referencia.**, see page **¡Error!Marcador no definido.**. The value of the year sent is the difference between the year the programming was created and the base year.

Command protected by a PIN.

Notes:

a) A NACK is returned if

- The Exit module does not have its software correctly loaded.
- The Validator does not have its tables correctly programmed.

5.36 REQUEST LAST MODIFICATION DATE [195]

This command requests the Validator for information about the date of the last modification of the programming of the coin tables in the Validator:

Send: [Dir] [0] [1] [195] [Chk]

Reply: [1] [2] [Dir] [0] [Data 1] [Data 2] [Chk] -> ACK with 2 data

Where Dir = address of the corresponding validator

Data 1 = LSB of the date.

Data 2= MSB of the date.

The format of date is the same that the indicated in the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if

- The Exit module does not have its software correctly loaded.
- The Validator does not have its tables correctly programmed.

5.37 REQUEST REJECT COUNTER [194]

This command requests the Validator for information about the coins rejected counter.

Send: [Dir] [0] [1] [194] [Chk]

Reply: [1] [3] [Dir] [0] [Data 1] [Data 2] [Data 3] [Chk] -> ACK with 3 data

Where Dir = address of the corresponding validator

Data 1 = number of coins rejected (LSB)

Data 2	= number of coins rejected (2SB)
Data 3	= number of coins rejected (MSB)

The validator keeps a counter with the number of coins rejected by different reasons than not being in the tables: inhibited coins, string detector mechanism activated, etc. This counter is of three bytes and can store up to 16.777.215 coins. It is incremented each time a coin is rejected by the Validator.

This counter works the same as the coins inserted counter. See the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

5.38 REQUEST FRAUD COUNTER [193]

This command requests the Validator for information about the coins rejected counter for not being in the coin tables (fraud attempt).

Send: [Dir] [0] [1] [193] [Chk]

Reply: [1] [3] [Dir] [0] [Data 1] [Data 2] [Data 3] [Chk] -> ACK with 3 data

Where Dir = address of the *corresponding validator*

Data 1	= number of coins rejected (LSB)
Data 2	= number of coins rejected (2SB)
Data 3	= number of coins rejected (MSB)

The validator keeps a counter with the number of coins rejected for not being in the coin tables: it is assumed that these coins are fraud attempts. This counter is of three bytes and can store up to 16.777.215 coins. It is incremented each time a coin that is not in the tables is rejected by the Validator.

This counter works the same as the coins inserted counter. See the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if the Exit module does not have its software correctly loaded

5.39 REQUEST BUILD CODE [192]

With this command, the corresponding Validator returns the Azkoyen reference of the device to the Machine.

Send: [Dir] [0] [1] [192] [Chk]

Reply: [1] [n] [Dir] [0] [Data 1] ... [Data 10] [Chk]

Where: Dir = address of the *corresponding validator*

Data 1 - Data 10 = string of charactersn = 10: Number of data to send

The Azkoyen reference consists of a string of 10 characters. This string is composed of 8 digits, followed by a dash '-' and a digit indicating the version.

The complete identification of the product can be determined using the commands **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.** and REQUEST BUILD CODE.

Command protected by a PIN.

Notes:

a) A NACK is returned if something of the following occurs:

- The Exit module does not have its software correctly loaded or does not have its tables correctly programmed

- The Sensor module does not answer, its software is not correctly loaded or its tables are not correctly programmed.

- The coin tables of the Exits module and the Sensor Module are not coincident or are not compatible.

Example:The Machine requests to the Validator number 2 its build code, which in this example is "41123121-0"

Send: [2] [0] [1] [192] [613]

Reply: [01] [10] [2] [00] ['4'] ['1'] ['1'] ['2'] ['3'] ['1'] ['2'] ['1'] ['-'] ['0'] [02]

Note: the reference "41123121-0" is only an example, is does not correspond to any real reference number.

5.40 MODIFY COIN ID [185]

With this command each coin is assigned an identity (6 characters maximum):

Send: [Dir] [7] [1] [185] [Data 1][char 1] [char 2] [char 3] [char 4] [char 5] [char 6] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the corresponding validator

Data 1 Position of the coin (1 a 16)

Data 2 … Data 6: ASCII characters of the coin identity

Each coin in the Validator is assigned a 6-character identity, which can be modified with this command and read using the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**. The identities are kept in the Flash memory of the Validator.

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded
- The Exit module does not have its tables correctly programmed.
- It has received an incorrect coin code
- There has been an error in the programming of the Flash memory of the Validator.

Example:

The Machine requests Validator number 2 to modify the identity of the coin of code 3 with the string “EU100”

Send: [2] [7] [1] [185] [3] [‘E’] [‘U’] [‘1’] [‘0’] [‘0’] [‘ ‘] [251]

Reply: [1] [1] [2] [0] [6] [246]

5.41 REQUEST COIN ID [184]

The Machine requests the Validator for the identity of the coin situated in the position indicated:

Send: [Dir] [1] [1] [184] [Data 1] [Chk]

Reply: [01] [06] [Dir] [00]][Data 2] [Data 3] [Data 4] [Data 5] [Data 6] [Data 7] [Chk] -> ACK with 6 data

Where Dir = address of the *corresponding validator*

Data 1 Position of the coin (1 a the 16)

Data 2 – Data 7: identity of the coin whose position is [Data 1]

The identity of the coin indicated is modified using the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

The number of characters returned is always 6:

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded
- The Exit module does not have its tables correctly programmed.
- It has received an incorrect coin code

5.42 MODIFY SECURITY SETTING [181]

The Machine defines for each coin position if the Validator uses a determined wide band, normal band, narrow band or super narrow band for a coin:

Send: [Dir] [2] [1] [181] [Data 1] [Data 2] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the *corresponding validator*

Data 1 Position of the coin (1 a the 16)

Data 2 = 0: normal band

= 1, 2, and 3: increase security.

= -1, -2, and -3: decrease security.

There is a compromise between security and percentage of admission of coins. Normally greater security corresponds to a better percentage of admission. The level of security is defined in the Validator in four levels or bands of acceptance for each coin. From greater to lesser security we have:

- Wide band, value -1: permits a greater percentage of admission of legal coins, normally used when there are no frauds for this coin.
- Normal band, value 0: it has the normal percentage of admission. It has good admission characteristics of legal coins: around 95% with good fraud rejection characteristics.
- Narrow band, value 1: reduces the percentage of admission of legal coins increasing the rejection of frauds.
- Super narrow band, value 2: maximum rejection of frauds is achieved although this is at the expense of the percentage of admission of legal coins.

The modification of the level of security or band of admission is carried out gradually indicating to the Validator how much it should increase or decrease the present level of security. It is necessary to take into consideration the following:

- The value 0 indicates to the Validator that the band of acceptance is “Normal Band”.
- If when increasing security a value superior to 2 is reached, super narrow band, the Validator takes the value 2.
- If when decreasing security a value inferior to -1 is reached, wide band, the Validator takes the value -1.

This information will be stored in the Flash memory of the Validator:

Command protected by a PIN.

Notes:

a) A NACK is returned if:
    - The Exit module does not have its software correctly loaded
    - The Exit module does not have its tables correctly programmed.
    - It has received an incorrect coin code
    - There has been an error to the programming of the Flash memory of the Validator.

5.43 REQUEST SECURITY SETTING [180]

The Machine requests the Validator for information about the type of band of acceptance or level of security with which the Validator is working for a determined coin:

Send: [Dir] [1] [1] [180] [Data 1] [Chk]

Reply: [1] [1] [Dir] [0] [Data 2] [Chk] -> ACK with a data

Where Dir = address of the *corresponding validator*

Data 1 Position of the coin (1 to 16)

Data 2 = -1: Wide band

0: Normal band (default)

1: Narrow band

2: Super narrow band

The modification process of security level and its characteristics is indicated in the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded
- The Exit module does not have their tables correctly programmed.
- It has received an incorrect coin code

5.44 MODIFY BANK SELECT [179]

The Machine indicates to the Validator which bank of coins is activated at each moment:

Send: [Dir] [1] [1] [179] [Data 1] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Where Dir = address of the corresponding validator.

Data 1 = 0, 1, 2, 3: Number of bank.

Only the coins that are assigned the indicated bank can be accepted.

This information will be stored in the Flash memory of the Validator:

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded
- The Exit module is programmed for work in mixed mode (Parallel and CCTALK).
- The Exit module does not have its tables correctly programmed.
- It has received an incorrect bank number
- There has been an error to the programming of the Flash memory of the Validator.

5.45 REQUEST BANK SELECT [178]

The Machine requests the Validator for the bank of coins that is activated:

Send: **[Dir] [0] [1] [178] [Chk]**

Reply: [1] [1] [Dir] [0] [Data 1] [Chk] -> ACK with a data

Where Dir = address of the corresponding validator.

Data 1 = 0, 1, 2, 3: Number of present bank

The bank of coins can be modified using the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded
- The Exit module is programmed for work in mixed mode (Parallel and CCTALK).
- The Exit module does not have its tables correctly programmed.

5.46 REQUEST ALARM COUNTER [176]

This command requests the Validator for information about the number of times that an alarm situation has occurred in the validator.

Send: [Dir] [0] [1] [176] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1] [Chk] -> ACK with a data

Where Dir = address of the corresponding validator

Data 1 = 0 – 255: Counter of the number of alarms generated from the last consult

The events defined as alarms are the following:

- Coin exit detectors continually covered.
- Measurement detectors continually covered.
- Coin rejected by the string detector system.
- Valid coin remains in the exit sensor for an excessive time.
- Valid coin does not reach the exit sensor.
- Valid coin reaches the exit sensor too soon.

The counter is kept in RAM and is reset each time the counter is read and after any reset or power-up.

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded

5.47 REQUEST BASE YEAR [170]

The Machine requests the Validator for information about the base year, from which the calculation of dates is carried out:

Send: [Dir] [0] [1] [170] [Chk]

Reply: [1] [4] [Dir] [0] [‘2’] [‘0’] [‘0’] [‘0’] [Chk] -> ACK with 4 data

Where Dir = address of the corresponding validator

The Validator it responds with the string “2000” as base year.

Command protected by a PIN.

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded

5.48 REQUEST ADDRESS MODE [169]

This command returns information on the cctalk addressing mode as help for the auto configuration of the different peripherals on the bus.

Send: [Dir] [0] [1] [169] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1] [Chk] -> ACK with a data

Where Dir = address of the corresponding validator

Data 1.Bit 0 = 0: Address saved in ROM (not used)

Bit 1 = 0: Address saved in RAM (not used)

Bit 2 = 1: Address saved in EEPROM or non Volatile Memory

Bit 3 = 0: Choice of address by the connector (not used)

Bit 4 = 0: Choice of address by jumpers on PCB (not used)

Bit 5 = 0: Choice of address by switch (not used)

Bit 6 = 0: the address can be modified by serial commands, volatile (not used)

Bit 7 = 1: the address can be modified by serial commands (not volatile)

The validator returns the value 132 (decimal) = 0x84 (hexadecimal) = 1000.0100 (binary)

Command protected by a PIN.

The Validator always responds to this command.

5.49 MODIFY INHIBIT AND OVERRIDE REGISTER [162]

This command permits the Machine to transmit information about the “present coin” and the “following coin” defined by each of its inhibit masks.

Send: [Dir] [6] [1] [162] [Data 1] [Data 2] [Data 3] [Data 4] [Data 5] [Data 6] [Chk]

Reply: [1] [0] [Dir] [0] [Chk]

Where Dir = address of the corresponding validator

Data 1 and Data 2: inhibit mask of the present coin.

Data 3: not used = 0.

Data 4 and Data 5: inhibit mask of the following coin.

Data 6: not used = 0.

If the "following coin" is always deactivated, the validator will only accept one coin at a time.

Command protected by a PIN.

a) A NACK is returned if the Exit module does not have its software correctly loaded or is programmed for work in mixed mode (parallel and CCTALK).

5.50 REQUEST FIRMWARE UPGRADE CAPABILITY [141]

Using this command the Validator informs the Machine where the firmware in the Validator is stored:

Send: [Dir] [0] [1] [141] [Chk]

Reply: [1] [1] [Dir] [0] [Data 1] [Chk] -> ACK with a data

Where Dir = address of the corresponding validator

Data = 1: Firmware in Flash memory

The Validator allows the remote modification of the firmware: see commands **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**, **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**, and **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Command protected by a PIN.

The Validator always responds to this command.

5.51 UPLOAD FIRMWARE [140]

Using this command the firmware of the Validator is modified:

Send: [Dir] [n] [1] [140] [Data 1] [Data 2] [Data 3] ... [Data n] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data.

Where Dir = address of the corresponding validator

Data 1 = block counter: not used

Data 2 = line counter: not used

Data 3 ... Data n = firmware data

The format of the data: Data 1 ... Data n, corresponds to that indicated in the CCTALK specifications level 1, version 4.2.

Azkoyen will provide the firmware data to transmit to the Validator: Data 3... Data n.

The data to transmit is variable. The maximum is 38 bytes in each string, the block and line counters are added to this.

The values corresponding to Data 1 and Data 2 (block and line counter) are not used by the Validator. The control of the integrity of the data is in the block of data provided by Azkoyen.

Command protected by a PIN.

a) A NACK is returned if:

- The command **¡Error!No se encuentra el origen de la referencia.**, see page **¡Error!Marcador no definido.** has not previously been transmitted.
- There has been an error in programming the Flash memory of the Validator.

b) While the firmware is being modified the majority of cctalk commands will return NACK indicating that the software is erroneous.

c) The reply time is 250 ms. maximum, typical: less than 50 ms.

5.52 BEGIN FIRMWARE UPGRADE [139]

This command is used to start the transmission of new firmware to the Validator:

Send:	[Dir] [0] [1] [139] [Chk]
Reply:	[1] [0] [Dir] [0] [Chk] -> ACK without data.

Where Dir = address of the corresponding validator

This command indicates to the Validator that the firmware is going to be modified. It is necessary to send this command first to modify the firmware of the Validator.

Command protected by a PIN.

a) A NACK is returned if the Validator has not been able to start the modification of the firmware.

5.53 FINISH FIRMWARE UPGRADE [138]

This command is used to finish the transmission of new firmware to the Validator:

Send:	[Dir] [0] [1] [138] [Chk]
Reply:	[1] [0] [Dir] [0] [Chk] -> ACK without data.

Where Dir = address of the corresponding validator

This command indicates to the Validator that the modification of the firmware has finished.

Command protected by a PIN.

a) A NACK is returned if the Validator has not been able to finish the modification of the firmware.

b) It may be necessary to modify the programming of coins of the Validator after transmitting the new firmware to the Validator.

5.54 SET ACCEPTANCE LIMIT [135]

Using this command the Machine tells the Validator how many coins must be accepted before it is auto-inhibited.

Send: [Dir] [1] [1] [135] [Data 1] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data.

Where Dir = address of the corresponding validator

Data 1 = maximum number of coins to be accepted (0, 1 ... 255)

The Validator will not accept any coin once it has reached the maximum number of accepted coins, indicated in [Data 1]. Every valid inserted coin will be rejected and the code returned will be 127+n (inhibited coin). See paragraph **¡Error!No se encuentra el origen de la referencia. ¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

If [Data 1] is equal to 0, the auto-inhibition will be deactivated and the Validator will accept coins in a continuous way. In this case, the commands that inhibit any coins individually or activate/deactivate the master inhibition must be used

After a reset the Validator will be in the same state as if this command with [Data 1] = 0 were received.

Command protected by a PIN.

a) A NACK is returned if the Validator does not have its software correctly loaded.

5.55 BEGIN TABLES UPLOAD [99]

Using this command the transmission of the data of programming of coins to the Validator is started:

Send: [Dir] [0] [1] [99] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data.

Where Dir = address of the corresponding validator

This command indicates to the Validator that the transmission of new programming of coins it is going to start.

Command protected by a PIN.

a) A NACK is returned if the Validator does not have its software correctly loaded or there has been an error in starting the modification of the tables of the Validator.

5.56 UPLOAD TABLES [98]

This command is used to modify the programming of coins of the Validator:

Send: [Dir] [n] [1] [98] [Data 1] [Data 2] [Data 3] ... [Data n] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data.

Where Dir = address of the corresponding validator

Data 1 = block counter: not used

Data 2 = line counter: not used

Data 3 ... Data n = coin programming data

The format of the data: Data 1 ... Data n is identical to that indicated in the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

Azkoyen will provide the coin programming data to transmit to the Validator: Data 3... to Data n.

The quantity of data to transmit is variable. The maximum is 36 bytes in each string, to that the block and line counters are added.

The values corresponding a Data 1 and Data 2 (block and line counters) are not used by the Validator. The control of the integrity of the data is in the block of data provided by Azkoyen.

Command protected by a PIN.

a) A NACK is returned if:

- The Validator does not have its software correctly loaded.
- The command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.** has not previously been transmitted.
- There has been an error in programming the Flash memory of the Validator.

b) After the coin tables of the Validator have been modified, any coin introduced will be rejected.

5.57 FINISH TABLES UPLOAD [97]

This command is used to finish the transmission of coin tables to the Validator:

Send: [Dir] [0] [1] [97] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data.

Where Dir = address of the corresponding validator

This command indicates to the Validator that the modification of the coin tables has finished.

Command protected by a PIN.

a) A NACK is returned if the Validator does not have its software correctly loaded or there has been an error in finishing the programming of the tables of the Validator.

5.58 REQUEST MODULES INFORMATION [96]

The Machine requests the Validator for information about certain information necessary for processing the firmware files and programming the tables of the Validator, provided by Azkoyen:

Send: [Dir] [0] [1] [96] [Chk]

Reply: [1] [51] [Dir] [0] [Data 1] ... [Data 51] [Chk] -> ACK with 51 data

Where Dir = address of the corresponding validator

Data 1... Data 51: Data with information on the Validator for the search for coin programming data and firmware in the files provided by Azkoyen.

The meaning of the data transmitted by the Validator is the following:

- Data 1: Type of Validator. The value transmitted by the A6-cctalk validator is the character ‘8’.
- Data 2: Type of sensor system the Sensor module has. The most usual values are the following:
  - ‘E’: High performance
  - ‘6’: High performance without sound sensor.
  - ‘1’: Normal performance.

The returned value in case of error can be:

  - ‘0’: The Sensor module does not respond
  - ‘F’: The Sensor module has its firmware incomplete.

- Data 3 – Data 16: Internal Azkoyen reference of the programming of coins of the Sensor module. It is a 14-character string of the type “332XXXXX-X,vXX”, where X are ASCII characters.

X = ‘0’ is returned if:

  - The Sensor module has its firmware incomplete, it is not correctly programmed or it does not respond.
  - The Exit module has its firmware incomplete.
- Data 17: Adjustments version of the coin programming of the Sensor module. It is a binary value. This value is 0xFF if:
  - The Sensor module has its firmware incomplete or it does not respond.
  - The Exit module has its firmware incomplete.
  - The Validator is not or bad programmed.
  - Adjustments version is not valid
- Data 18 – 31: Configuration Name of the Exit module. "00000000-0,v00" is returned if:
  - The Sensor module has its firmware incomplete or it does not respond.
  - The Exit module has its firmware incomplete.
  - The Validator is not or bad programmed.
  - Configuration data are not valid
- Data 32 – 45: String of 14 chars:
  - "00000000-0,v00" if something of the following occurs: The Sensor module has its firmware incomplete or it does not respond, the Exit module has its firmware incomplete, the Validator is not or bad programmed or the data configuration are not valid. In anyone of these cases the Validator can not admit any coin.
  - "332XXXXX-X,vXX" if the Sensor and Exit modules do not have a compatible coin programming. This string corresponds to the coin programming reference of the Exit module. In this case the Validator could work although with a reduced performance. The error code returned can be 6 or 7 (see **¡Error!No se encuentra el origen de la referencia.**).
  - "41XXXXXX-X,vXX" if the coin programming in both modules are correct an coincident. The Validator is working normally.
- Data 46: Module number connected. If the Sensor module has responded, the value returned is 2. If the Sensor module has not, it responds with the value 1.
- Data 47: Firmware version of the Sensor module. It is a binary value from 1 to 255. In case of error the value can be one of the following:
  - 0: The Sensor module does not respond

  - 255: The firmware of the Sensor module is incomplete.

- Data 48: Firmware version of the Exit module. It is a binary value from 1 to 255.
- Data 49 and 50: Reserved for future use.
- Data 51: Error code resulting from the comparison of the programming coin characteristics between the Sensor and Exit modules. See **¡Error!No se encuentra el origen de la referencia.**.

Código	Descripción
0	Sin errores
1	Módulo Sensor no conectado o no funciona
2	Firmware Módulo de Salidas incorrecto
3	Firmware Módulo Sensor incorrecto
4	Selector (Modulo de Salidas o Módulo Sensor) con programación de monedas incorrecta o no programado.
5	Datos de Configuración (usuario) del Módulo de Salidas con valores incorrectos
6	Indicativos de países de la programación de monedas no son compatibles
7	Referencias de prog. de monedas del Módulo Sensor y Módulo de Salidas no coinciden

Table ¡Error!Argumento de modificador desconocido.**: Command "Request Modules Information": error codes**

The data received by the Validator is used by the corresponding programming tools to obtain the coin programming data files and the firmware data files.

The coin programming data is unique for each Validator. Azkoyen provides files that have the data of various validators. To find the data to transmit in this file it is necessary to have the data received with this command such as the serial number of the Validator, see command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

To get the data referring to the new firmware, it is necessary to have the data received using this command: REQUEST MODULES INFORMATION [96].

Command protected by a PIN.

The Validator always responds to this command.

The maximum time of reply is 1.2 seconds if the Sensor module is disconnected or it does not work correctly. Typical: 20 ms.

5.59 REQUEST COMMS REVISION [4]

As a reply to this command, the Validator returns the level of implementation of the cctalk protocol® and the version of the communication software:

Send: [Dir] [0] [1] [4] [Chk]

Reply: [1] [3] [Dir] [0] [Data 1] [Data 2] [Data 3] [Chk]

Where: Data 1 = Level of the communication protocol

Data 2 = Major revision

Data 3 = Minor revision

Dir = address of the corresponding validator

The version presently implemented is:

Level = 1

Major revision = 4

Minor revision = 2

Command protected by a PIN.

The Validator always responds to this command.

5.60 CLEAR COMMS STATUS VARIABLES [3]

Reset to zero all the counters named in the command **¡Error!No se encuentra el origen de la referencia.**, see page **¡Error!Marcador no definido.**:

Send: [Dir] [0] [1] [3] [Chk]

Reply: [1] [0] [Dir] [0] [Chk] -> ACK without data

Command protected by a PIN.

The Validator always responds to this command.

5.61 REQUEST COMMS STATUS VARIABLES [2]

This command requests the Validator for information about the counters controlling the quality of the communication.

Send: [Dir] [0] [1] [2] [Chk]

Reply: [1] [3] [Dir] [0] [Data 1] [Data 2] [Data 3] [Chk] -> ACK with 3 data

Where	Dir = address of the corresponding validator
	Data 1 = Rx_timeouts
	Data 2 = Rx_bytes_ignored
	Data 3 = Rx_bad_checksum

- Rx_timeouts: this counter accumulates the number of times the validator gives a time-out in the reception of data. If the communication is carried out correctly this value should be as close to zero as possible

- Rx_bytes_ignored: if the machine sends the validator very long messages that are longer than the length of the validator buffer and they are lost, the number of lost bytes is added to this counter.

- Rx_bad_cheksum: this counter is incremented each time that a message with an erroneous checksum is received.

Each time that a reset or power-up is done, the validator counters will start at 0. Before using and evaluating the data received with this command it is advised to first execute the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

The parameters returned by this command give an idea of the noise level in the communication and how well it is being established.

5.62 RESET DEVICE [1]

This command makes the Validator carry out a hardware reset. At the moment of execution the Validator programme goes into the reset vector. The affected Validator sends an ACK string immediately before carrying out the reset.

Send:	[Dir] [0] [1] [1] [Chk]
Reply:	[1] [0] [Dir] [0] [Chk] -> ACK without data

All the sorter coin and path counter information assigned to each coin and the configuration of inhibit bits is lost.

6. TELE-PROGRAMMING PROCESS

The coin data stored in the Validator is individualised to account for the manufacturing tolerances. It has a coin data programming system for programming in the field (tele-programming) which uses the measurement characteristics (calibration) of each Validator and that is stored in a data base in the Factory.

To create or modify the programming of a determined Validator the serial number of this Validator is required. With this number it is possible to obtain the data to transmit to the Validator. Azkoyen supplies a file that contains the coin data of various validators. The format of this file is indicated in the specifications.

To create the tele-programming file and find the data corresponding to the Validator installed in the Machine it is necessary obtain the following information from the Validator:

- Serial number: use the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.
- Information on the modules that the Validator is made up of: command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**.

The data to transmit to the Validator consists of a set of strings with the data that should be transmitted to the validator. The format of this data is as follows:

[n][cctalk command n][Data 1n]... [Data n]: string 1

[m][cctalk command m][Data 1m]... [Data m]: string 2

...

[x][cctalk command x][Data 1x]... [Data x]: string z

Each string has:

- [n]: number of bytes to transmit to the Validator
- [cctalk command n]: cctalk command to transmit to the Validator
- [Data 1]... [Data n]: data to transmit to the Validator.

The file is binary. All the strings are contiguous, that is, after the byte [Data n] will be the data [m].

The Machine should generate the CCTALK string adding the header, the command and data indicated in the string and calculate the corresponding checksum:

[Dir Validator]: address of the Validator

[n]: number of bytes indicated in the string of the binary file

[Dir Machine]: address of the Machine

[cctalk command n]: command indicated in the string of the binary file

[Data 1n]... [Data n]: data indicated in the string of the binary file

[Checksum]: checksum calculated by the Machine the CCTALK string.

This string will be transmitted to the Validator. If it responds ACK, it will create the following string waiting for the corresponding reply. The process will be repeated until all the data is transmitted or an error occurs, the Validator does not respond or it responds with a NACK.

Notes:

a) If an error occurs in the Tele-programming process the Validator will be incorrectly programmed so that it will not be capable of accepting coins. It is necessary to repeat the process again so that the Validator is correctly programmed.

7. FIRMWARE PROGRAMMING PROCESS

When it is necessary to modify the firmware of the Validator (the Sensor module and the Exit module) due to updates or error corrections, Azkoyen will provide a file with the data to transmit to the Validator. To locate this file with the data to transmit to the Validator the information returned from the command **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.** is necessary. The format of the file is indicated in the specifications.

The data corresponding to the Validator and that has been obtained from the file have the same format as indicated in **¡Error!No se encuentra el origen de la referencia.**, page **¡Error!Marcador no definido.**. The process of creating CCTALK strings is the same as indicated in the previous section.

Notes:

a) While transmitting the new firmware the power supply of the Validator should remain stable. The programming process is prepared for cuts in the communication or in the power supply so the Validator does not go out of order and the process can be repeated. However, there is a risk of leaving the Validator permanently out of order if the power supply is removed or suffers continuous interruptions during the programming of the firmware process.

b) If there is an error in programming the firmware and the Validator is only partially programmed and does not respond correctly to the majority of the cctalk commands, the process should be repeated from the beginning.

c) It will not usually be necessary to modify the programming of the coin data of the Validator after modifying the firmware. Azkoyen will indicate if it is necessary to carry out the Tele-programming of the Validator or not.