



# **MODULAR X6 CCTALK**



## INDEX

1.	CCTALK® SERIAL COMMUNICATION PROTOCOL .....	6
1.1	Introduction .....	6
2.	PHYSICAL LEVEL .....	8
2.1	Voltage levels and transmission speeds .....	8
2.2	Connector .....	8
3.	LOGIC LEVEL .....	10
3.1	Message structure .....	10
3.2	Destination address .....	10
3.3	Number of data bytes .....	11
3.4	Origin address .....	11
3.5	Header .....	11
3.6	Data .....	12
3.7	Checksum .....	12
3.8	Temporary requirements .....	12
4.	START UP PROCEDURE .....	13
5.	LIST OF COMMANDS .....	14
5.1	SIMPLE POLL [254] .....	20
5.2	ADDRESS POLL [253] .....	20
5.3	ADDRESS CLASH [252] .....	21
5.4	ADDRESS CHANGE [251] .....	22
5.5	ADDRESS RANDOM [250] .....	23
5.6	REQUEST POLLING PRIORITY [249] .....	24
5.7	REQUEST STATUS [248] .....	24
5.8	REQUEST MANUFACTURER ID [246] .....	25
5.9	REQUEST EQUIPMENT CATEGORY ID [245] .....	26
5.10	REQUEST PRODUCT CODE [244] .....	26
5.11	REQUEST DATABASE VERSION [243] .....	27

---



5.12	REQUEST SERIAL NUMBER [242] .....	28
5.13	REQUEST SOFTWARE REVISION [ 241] .....	28
5.14	TEST SOLENOIDS [240] .....	29
5.15	READ INPUT LINES [237] .....	30
5.16	READ OPTO STATES [236] .....	31
5.17	PERFORM SELF-CHECK [232] .....	32
5.18	MODIFY INHIBIT STATUS [231] .....	35
5.19	REQUEST INHIBIT STATUS [230] .....	36
5.20	READ BUFFERED CREDIT OR ERROR CODES [229] .....	37
5.21	MODIFY MASTER INHIBIT STATUS [228] .....	40
5.22	REQUEST MASTER INHIBIT STATUS [227] .....	41
5.23	REQUEST INSERTION COUNTER [226] .....	42
5.24	REQUEST ACCEPT COUNTER [225] .....	43
5.25	MODIFY SORTER OVERRIDE STATUS [222] .....	44
5.26	REQUEST SORTER OVERRIDE STATUS [221] .....	45
5.27	ENTER A NEW PIN NUMBER [219] .....	46
5.28	ENTER A PIN NUMBER [218] .....	47
5.29	REQUEST DATA STORAGE AVAILABILITY [216] .....	48
5.30	READ DATA BLOCK [215] .....	49
5.31	WRITE DATA BLOCK [214] .....	50
5.32	REQUEST OPTION FLAGS [213] .....	51
5.33	REQUEST COIN POSITION [212] .....	51
5.34	MODIFY SORTER PATHS [210] .....	52
5.35	REQUEST SORTER PATHS [209] .....	55
5.36	TEACH MODE CONTROL [202] .....	56
5.37	REQUEST TEACH STATUS [201] .....	57
5.38	CONFIGURATION TO EEPROM [199] .....	58
5.39	COUNTERS TO EEPROM [198] .....	59



5.40	CALCULATE ROM CHECKSUM [197]	60
5.41	REQUEST CREATION DATE [196]	61
5.42	REQUEST LAST MODIFICATION DATE [195]	62
5.43	REQUEST REJECT COUNTER [194]	62
5.44	REQUEST FRAUD COUNTER [193]	63
5.45	REQUEST BUILD CODE [192]	64
5.46	MODIFY DEFAULT SORTER PATH [189]	65
5.47	REQUEST DEFAULT SORTER PATH [188]	66
5.48	MODIFY COIN ID [185]	67
5.49	REQUEST COIN ID [184]	68
5.50	MODIFY SECURITY SETTING [181]	69
5.51	REQUEST SECURITY SETTING [180]	70
5.52	MODIFY BANK SELECT [179]	71
5.53	REQUEST BANK SELECT [178]	72
5.54	REQUEST ALARM COUNTER [176]	72
5.55	REQUEST BASE YEAR [170]	73
5.56	REQUEST ADDRESS MODE [169]	74
5.57	MODIFY INHIBIT AND OVERRIDE REGISTER [162]	75
5.58	REQUEST FIRMWARE UPGRADE CAPABILITY [141]	75
5.59	UPLOAD FIRMWARE [140]	76
5.60	BEGIN FIRMWARE UPGRADE [139]	77
5.61	FINISH FIRMWARE UPGRADE [138]	78
5.62	SET ACCEPTANCE LIMIT [135]	78
5.63	BEGIN TABLES UPLOAD [99]	79
5.64	UPLOAD TABLES [98]	80
5.65	FINISH TABLES UPLOAD [97]	81
5.66	REQUEST MODULES INFORMATION [96]	81
5.67	REQUEST COMMS REVISION [ 4 ]	84



5.68	CLEAR COMMS STATUS VARIABLES [ 3 ] .....	85
5.69	REQUEST COMMS STATUS VARIABLES [ 2 ] .....	85
5.70	RESET DEVICE [ 1 ] .....	87
6.	TELE-PROGRAMMING PROCESS .....	87
7.	FIRMWARE PROGRAMMING PROCESS .....	89



## 1. CCTALK® SERIAL COMMUNICATION PROTOCOL

This technical manual comprises the serial communication protocol cctalk® implemented on the X6 validators.

### 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").
- The validator can work with a 5 or 3 ways sorter.
- The default sorter path can be programmed with the command "Modify default sorter path".
- The sorter paths can be inhibited: command "Modify sorter override status". Each coin has programmed 4 possible sorter paths. If the path programmed as a first option (path 1) is inhibited the coin will be diverted to path 2. If this is inhibited will be diverted to path 3 and if it is inhibited the coin will go to path 4. If all the paths are inhibited the coin will go to the default path.
- 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 the possibility to work in "secure mode": certain commands are not admitted in order to avoid the modification of the working way of the Validator.



Additionally, the validator can work in mode "L66 cctalk compatibility". In this mode some commands have changed its behaviour in order to be compatible with the L66 cctalk validator.

- 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.c7.m0.x8.e0.i3.r4: conector de tipo 7.**



## 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 two connectors:

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

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

**Table 1: 4 pins Conector CCTALK**



10 pins connector, equivalent to Molex serial 862, P/N 10-89-1101 (10 $\mu$  gold). The description is as follows:

PIN	NAME	DESCRIPTION
1	/DATA	Data line
2	N.C.	Not connected
3	S1	Sorter coil 1
4	S3	Sorter coil 3
5	N.C.	Not connected
6	S2	Sorter coil 2
7	VIN	Power supply of the validator
8	GND	Ground, internally connected to 0V
9	N.C.	Not connected
10	N.C.	Not connected

**Table 2: 10 pins CCTALK connector**



**Nota 1:** This connector is designed to work with a sorter of 3 or 5 ways.



### 3. LOGIC LEVEL

#### 3.1 Message structure

The format of the messages is the following:



**[Destination address]**

**[N° 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 X6 Validator has the capacity to receive up to a maximum of 66 bytes in each string. In reception, it can receive and calculate the checksum of a string up to 252 bytes, although it can only process the first 66 (see command REQUEST COMMS STATURS VARIABLES [2]).

### 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 **ENTER PIN NUMBER [218]**. This command is only necessary if a PIN protection is used.



- Command **REQUEST STATUS [248]**: if the reply is not OK the Validator will not be able to accept coins. Optionally the command **PERFORM SELF-CHECK [232]** can be used.
- Command **MODIFY INHIBIT STATUS [231]**: Activate the coins that should be admitted.
- Command **MODIFY SORTER OVERRIDE STATUS [222]**. This command tells the validator the available sorter paths. It is not necessary to send this command if the sorter is not utilised. In this case, it is assumed that the coins have not programmed their own sorter path.
- Command **READ BUFFERED CREDIT OR ERROR CODES [229]**. 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 Vaidators.

A PIN can protect all the commands, except "Address Poll (253)" and "Simple Poll (254)". 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.

The X66 validator can work in one of the 3 following modes:

- Normal mode: the majority of the cctalk commands are available. See Table 3, page 11.
- Secure mode: some commands are not available. See Table 4, page 13.
- "L66 compatibility" mode: it is similar to "Secure Mode", some commands are not available (see Table4, page 17). Additionally, some others commands have their behaviour modified or are not available. See Table, page 14 .

**Table 1: LIST OF COMMANDS**

<b>Header</b> (decimal)	<b>Command</b>	<b>Page</b>
254	Simple Poll	19
253	Address Poll	20
252	Address Clash	21
251	Address Change	22
250	Address Random	23
249	Request polling priority	23
248	Request status	24
246	Request manufacturer id	25
245	Request equipment category id	26
244	Request product code	26
243	Request database version	27
242	Request serial number	27
241	Request Software Revision	28
237	Read Input lines	29
240	Test Solenoids	30
236	Read opto states	31
232	Perform self-check	32
231	Modify inhibit status	<b>¡Error! Marcador no definido.</b>
230	Request inhibit status	36



229	Read buffered credit or error codes	37
228	Modify master inhibit status	40
227	Request master inhibit status	41
226	Request insertion counter	41
225	Request accept counter	43
222	Modify sorter override status	43
221	Request sorter override status	45
219	Enter new a PIN number	46
218	Enter a PIN number	47
216	Request data storage availability	47
215	Read data block	48
214	Write data block	49
213	Request option flags	50
212	Request coin position	51
210	Modify sorter paths	52
209	Request sorter paths	54
202	Teach mode control	56
201	Request teach status	57
199	Configuration to EEPROM	58
198	Counters to EEPROM	59
197	Calculate ROM checksum	60
196	Request creation date	60
195	Request last modification date	61
194	Request reject counter	62
193	Request fraud counter	63
192	Request build code	64
189	Modify default sorter path	65



188	Request default sorter path	66
185	Modify coin id	67
184	Request coin id	68
181	Modify security setting	69
180	Request security setting	70
179	Modify bank select	71
178	Request bank select	72
176	Request alarm counter	72
170	Request base year	73
169	Request address mode	74
162	Modify inhibit and override registers	75
141	Request firmware upgrade capability	75
140	Upload firmware	76
139	Begin firmware upgrade	77
138	Finish firmware upgrade	78
135	Set acceptance limit	78
99	Begin tables upload	79
98	Upload tables	80
97	Finish tables upload	81
96	Request modules information	81
4	Request comms revision	84
3	Clear comms status variables	85
2	Request comms status variables	85
1	Reset device	87

**Table 4: Commands not accepted in "Secure Mode"**

Header	Command	Page
--------	---------	------



(decimal)		
216	Request data storage availability	47
215	Read data block	48
214	Write data block	49
202	Teach mode control	56
201	Request teach status	57
185	Modify coin id	67
181	Modify security setting	69
180	Request security setting	70
179	Modify bank select	71
178	Request bank select	72
140	Upload firmware	76
139	Begin firmware upgrade	77
138	Finish firmware upgrade	78
99	Begin tables upload	79
98	Upload tables	80
97	Finish tables upload	81

**Table 5: Modified commands in "L66 compatibility" mode**

<b>Header</b> (decimal)	<b>Command</b>	<b>Observations</b>	<b>Page</b>
248	Request status	Not available	24
237	Read Input lines	Available only in "L66 compatibility" mode	30
236	Read opto states	Funcionamiento modificado	<b>¡Error! Marcador</b>



			<b>no defin ido.</b>
228	Modify master inhibit status	Not available	40
227	Request master inhibit status	Not available	41
222	Modify sorter override status	Not available	43
221	Request sorter override status	Not available	45
213	Request option flags	Not available	50
212	Request coin position	Not available	51
199	Configuration to EEPROM	Not available	58
198	Counters to EEPROM	Not available	59
197	Calculate ROM checksum	Not available	60
196	Request creation date	Not available	60
195	Request last modification date	Not available	60
194	Request reject counter	Not available	61
193	Request fraud counter	Not available	62
189	Modify default sorter path	Not available	63
188	Request default sorter path	Not available	65
176	Request alarm counter	Not available	66
170	Request base year	Not available	72
162	Modify inhibit and override registers	Not available	73
135	Set acceptance limit	Not available	75
3	Clear comms status variables	Not available	78
2	Request comms status variables	Not available	85



## 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.

### 5.3 ADDRESS CLASH [252]

This command is used to check if one or more devices share the same address. The difference to the command ADDRESS POLL [253], 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  $254 \times 256 = 1$  in 65.024).

The Validator always answers 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.

## 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 SIMPLE POLL [254].
- c) If the Validator cannot save the new value of the address in its memory a NACK is returned.



d) If the new address is 0 or 1 the Validator takes the default address (2).

**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 SIMPLE POLL [254] 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 READ BUFFERED CREDIT OR ERROR CODES [229]).

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.

Notes:

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

**5.7 REQUEST STATUS [248]**

This command is not available if the Validator is programmed in "L66 compatibility" mode

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 Table 6

**Table 6: Status codes**

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

Command protected by a PIN.

Notes:

- b) This command will not carry out a complete check of the validator. For this use the command PERFORM SELF-CHECK [232], is recommended.
- c) If the Validator does not have its software correctly loaded, a NACK is returned to this command.
- d) The Validator does not answer to this command if it is programmed in "L66 compatibility" mode.

### **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 REQUEST BUILD CODE [192].

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 ('X6': 2 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 ("X6") 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 be 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.
- b) If the Validator is working in "Secure Mode" or in "L66 compatibility" mode, it returns 0 indicating that the teleprogramming process is not possible.



### 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 3 bytes:

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

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

Where Dir = address of the Validator.

LSB, 2SB, MSB: bytes of the serial number (3 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 REQUEST PRODUCT CODE [244] returns, followed by the string " cctalk".
- 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 REQUEST PRODUCT CODE [244].

Command protected by a PIN.

**Example:**

The Machine requests *Validator* number 2 for its software version, which in this example is 'X6 cctalk V1.2'.

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

Reply: [1] [14] [2] [0] ['X'] ['6'] [' ' ] ['c'] ['c'] ['t'] ['a'] ['l'] ['k'] [' ' ] ['V'] ['1'] ['.' ]  
[ '2' ] [chk]

### 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

Bit 1 – Sorter solenoid 1

Bit 2 – Sorter solenoid 2

Bit 3 – Sorter solenoid 3



Each solenoid is activated for 500 ms. if the corresponding bit is at "1". The ACK reply of the Validator is produced after the end of the command, 2 seconds maximum in depends of the number of solenoids to activate. Each solenoid is activate sequentially if several are requested.

Command protected by a PIN.

Notes:

a) A NACK is returned if:

The Exit module does not have its software correctly loaded.

The argument: [Data 1] is greater than 16 (0x0F).

There is a coin inside or the Validator is activating the exits or the solenoids at the moment of reception of the command.

b) 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 1 sec. (500 ms. each solenoid)

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

**5.15 READ INPUT LINES [237]**

This command is only available if the Validator is programmed in "L66 compatibility" mode.

The L66-cctalk validator returns the switches status and the string detector mechanism.

Send: [Dir] [0] [1] [237] [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 – Bit 3: always at 0 (is equivalent to closed switches)

Bit 4: String detector mechanism (0 = closed / 1 = open)

The X6 validator does not have any switches, thus it returns always 0: closed switches

Command protected by a PIN.

Notes:

- a) A NACK is returned if the Exit module does not have its software correctly loaded.
- b) The X6 validator does not answer to this command if it is not programmed in "L66 compatibility" mode.

## 5.16 READ OPTO STATES [236]

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

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 = 0

Bit 1: 0 if all of the measurement fototransistors of the Sensor Module are free.

1 if some of this fototransistors are blocked.

Bit 2: Coin exit phototransistor

Bit 3: String detector mechanism (if the Validator is not programmed in "L66 compatibility" mode).

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.

The Bit 3 is only used if the Validator is NOT programmed in "L66 compatibility" mode. The L66-cctalk Validator does not make any test of the string detector in this command. It always send 0.

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.

## 5.17 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 Table 7.

**Table 7: Hardware Test – Error Codes**

<b>Code</b>	<b>Type of Error [Data 1]</b>	<b>Complementary Information [Data 2]</b>
0	OK (no fault has been detected )	-
1	Flash memory corrupted	-
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	Bit 0: sensor FT0 Bit 1: sensor FT1 Bit 3: sensor FT2
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	-

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 codes 2 and 6 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

## 5.18 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.
- 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 REQUEST INHIBIT STATUS [230], 37.

**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]

## 5.19 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 MODIFY INHIBIT STATUS [231].

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 MODIFY INHIBIT STATUS [231].

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. The value of the inhibit bits can be modified by the means of the command MODIFY INHIBIT STATUS [231].

***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.

## **5.20 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 ( $x = 1, 2, 3, 4, \text{ and } 5$ ):

- 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 Table 8.  
= 0 if there is not an error.

In Table 8 the error codes that give information on the possible causes of the rejection of the coin are presented.

**Table 8: Command Read Buffered Credit: Error codes**

<b>Code</b>	<b>Error</b>
0	Empty event (no error)
1	Coin Rejected (Sensor module is not correctly programmed, the coin corresponds to a wider acceptance band than the expected or not valid coin: fraud attempt)
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
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 REQUEST POLLING PRIORITY [249]. 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.

## **5.21 MODIFY MASTER INHIBIT STATUS [228]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

## 5.22 REQUEST MASTER INHIBIT STATUS [227]

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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 MODIFY MODIFY MASTER INHIBIT STATUS [228].

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: normal working mode, after a reset or power-up.
- c) The Validator does not answer if it is programmed in "L66 compatibility" mode.

### 5.23 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 COUNTERS TO EPROM [189], 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 COUNTERS TO EEPROM [198] 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.

## 5.24 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 REQUEST INSERTION COUNTER [226].

Command protected by a PIN.



Notes:

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

## 5.25 MODIFY SORTER OVERRIDE STATUS [222]

This command is not available if the Validator is programmed in "L66 compatibility" mode.

The machine indicates to the validator that classification paths are permitted. Each bit represents a sorter path.

Send: [Dir] [1] [1] [222] [Data 1] [Chk]

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

Where Dir = address of the corresponding validator

Data 1 = a set of bits that indicates the path permitted:

Bit 0 = Path 1

Bit 1 = Path 2

Bit 2 = Path 3

Bit 3 = Path 4

Bit 4 = Path 5

Bit = 1 = without action. Normal classification.

Bit = 0 = this path is not permitted. The coin is sent via another path or to the default path.

This value is normally saved in the Flash memory of the Validator and is manipulated in RAM. If the command CONFIGURATION TO EEPROM [199], is not used all the paths are prohibited. All the coins will go to the default path. However, if the command CONFIGURATION TO EEPROM [199] is used, the value that it has at that moment is saved to the Flash memory of the Validator and the next power-up the value in the Flash memory is read.

The present value of the permitted classification paths can be read from the Validator using the command REQUEST SORTER OVERRIDE STATUS [221].



The value of the default classification path is indicated to the Validator using the command MODIFY DEFAULT SORTER PATH [189].

Command protected by a PIN.

Notes:

- a) A NACK is returned if the Exit module does not have its software correctly loaded..
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

**Example:**

The Machine sends the following information about the paths permitted of classification to Validator number 2:

- Paths 1 and 3: permitted
- Rest of paths: prohibited:

The value transmitted will be (in binary): 0000.0101 = 5 (decimal)

Send: [2] [1] [1] [222] [5] [25]

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

**5.26 REQUEST SORTER OVERRIDE STATUS [221]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

The Machine requests the Validator for information about the classification paths that are permitted.

Send: [Dir] [0] [1] [221] [Chk]

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

Where Dir = address of the corresponding validator

Data 1 = a set of bits that indicates the path permitted:

Data 1. Bit0 = Path 1

Data 1. Bit1 = Path 2

.....



Data 1. Bit7 = Path 8

Bit = 1: path permitted.

Bit = 0: path prohibited.

These values can be modified using the command MODIFY SORTER OVERRIDE STATUS [222].

Command protected by a PIN.

Notes:

- a) A NACK is returned if the Exit module does not have its software correctly loaded.
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

**Example:**

The Machine requests Validator number 2 for the classification paths that are permitted and the paths that are prohibited:

Send: [2] [0] [1] [221] [32]

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

The value received is (in binary): 0000.0110 = 6 (decimal)

- Paths 2 and 3: permitted
- Rest of paths: prohibited:

**5.27 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 SIMPLE POLL [254]) 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 ENTER A PIN NUMBER [218]).

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.

## 5.28 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 SIMPLE POLL [254]) 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.



## 5.29 REQUEST DATA STORAGE AVAILABILITY [216]

Command not available in "Secure Mode".

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.

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

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

Data 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 5.30- READ DATA BLOCK [215], and 5.31- WRITE DATA BLOCK [214], 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
- b) This command is not available if the Validator is working in "Secure Mode".

### 5.30 READ DATA BLOCK [215]

Command not available in "Secure Mode".

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.

b) This command is not available if the Validator is working in "Secure Mode".

### 5.31 WRITE DATA BLOCK [214]

Command not available in "Secure Mode".

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".

[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.

b) This command is not available if the Validator is working in "Secure Mode".

### 5.32 REQUEST OPTION FLAGS [213]

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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 REQUEST COIN POSITION [212], and REQUEST INHIBIT STATUS [230]).

Command protected by a PIN.

Notes:

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

b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

### 5.33 REQUEST COIN POSITION [212]

This command is not available if the Validator is programmed in "L66 compatibility" mode.



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 READ BUFFERED CREDIT OR ERROR CODES [229].
- Inhibit word: value of the coin inhibits – coins to admit or reject, used in the commands REQUEST INHIBIT STATUS [230], and MODIFY INHIBIT STATUS [231].

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)

b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

### **5.34 MODIFY SORTER PATHS [210]**

This command is used to configure in the Validator the sorter path that each coin is assigned. Two formats of the command are admitted:

**Format (a):**

Send: [Dir] [2] [1] [210] [Data 1] [Data 2] [Chk]

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



Where Dir = address of the corresponding *Validator*

[Data 1] = Code of coin

[Data 2] = Path of sorter assigned.

If the assigned classification path (Data 2) is prohibited the coin will go to the default path.

**Format (b):**

Send: [Dir] [5] [2] [210] [Data 1] [Data 2] [Data 3] [Data 4] [Data 5] [Chk]

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

Where Dir = address of the corresponding *Validator*

[Data 1] = Code of coin

[Data 2] = Assigned sorter path 1.

[Data 3] = Assigned sorter path 2.

[Data 4] = Assigned sorter path 3.

[Data 5] = Assigned sorter path 4.

The priority of the classification paths is:

Path 1 (Data 2) (maximum priority)

Path 2 (Data 3) (second priority): if path 1 is prohibited

Path 3 (Data 4) (third priority): if path 2 is prohibited

Path 4 (Data 5) (minimum priority): if path 3 is prohibited

If path 4 is prohibited the coin is sent to the default path.

In both formats, the value of path equal to 0 means default path. The values to transmit have the following meanings:

0: default path (see command MODIFY DEFAULT SORTER PATH [189],



- 1: sorter path 1
- 2: sorter path 2
- 3: sorter path 3
- 4: sorter path 4
- 5: sorter path 5

To permit or prohibit a determined path the command MODIFY SORTER OVERRIDE STATUS [222], is used.

The value of the classification of each coin is saved in the Flash memory of the Validator. These values are originally set to the value of the default path (0).

Command protected by a PIN.

Notes:

- a) A NACK is returned if:
  - The Exit module does not have its software correctly loaded.
  - Any of the calling arguments: [Data 1] to [Data 5] have an incorrect value (greater than 5).
  - There has been an error in programming the Flash memory of the Validator.
- b) In "L66 compatibility" mode, the default path is always channel 1.

**Example:**

Format (a):

The Machine configures Validator number 2 so that coin 6 is sent to sorter path 1. If this path is busy it should be sent to the default path.

Send: [2] [2] [1] [210] [6] [1] [34]

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

Format (b):

---



The Machine configures the Validator number 2 so that coin 6 is sent to sorter path 1. If it is busy it should leave through path 2 and if this is also busy it should leave through path 3. If this is busy it should be sent to the default path.

Send: [2] [5] [1] [210] [6] [1] [2] [3] [0] [26]

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

### 5.35 REQUEST SORTER PATHS [209]

This command requests the Validator for information about the assigned sorter path of a coin. The answer depends on the Validator working mode:

“Normal” working mode:

Send: [Dir] [1] [1] [209] [Data 1] [Chk]

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

Where Dir = address of the corresponding *Validator*

[Data 1] = Code of coin

[Data 2] = Assigned sorter path 1.

[Data 3] = Assigned sorter path 2.

[Data 4] = Assigned sorter path 3.

[Data 5] = Assigned sorter path 4.

“L66 compatibility” working mode:

Send: [Dir] [1] [1] [209] [Data 1] [Chk]

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

Where Dir = address of the corresponding *Validator*

[Data 1] = Code of coin

[Data 2] = Assigned sorter path.

The values returned are the programmed using the command MODIFY SORTER PATHS [210].

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 assigned sorter path of a coin with code 6.

Send: [2] [1] [1] [209] [6] [36]

Reply: [1] [1] [2] [0] [4] [3] [2] [1] [chk]

From this reply we deduce that coin 6 has the assigned sorter path number 4. If this path is not permitted the coin will be deviated to path 3, path 2 and path 1 respectively.

### 5.36 TEACH MODE CONTROL [202]

Command not available in "Secure Mode".

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 25<sup>th</sup> coin. The coins must not have defects such as dents or 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 REQUEST TEACH STATUS [201].

Command protected by a PIN.

Notes:

- 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.
- BUSY is returned (6) if the Validator is busy measuring or accepting a coin at the moment of receiving the command.
- This command is not available if the Validator is working in "Secure Mode".

### 5.37 REQUEST TEACH STATUS [201]

Command not available in "Secure Mode".

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 Table 9:

**Table 9: 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.
- c) This command is not available if the Validator is working in "Secure Mode".

### **5.38 CONFIGURATION TO EEPROM [199]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

The Machine requests the Validator to store the configuration data in the non volatile memory of the Validator: in this case, the Flash memory.

Send: [Dir] [0] [1] [199] [Chk]

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



Only the configuration of paths permitted and prohibited is implemented at the moment. See command MODIFY SORTER OVERRIDE STATUS [222].

This configuration will be read after a reset or power-up. If this command is not used, the configuration data corresponds to the default data: all the paths are prohibited.

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.

b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

### **5.39 COUNTERS TO EEPROM [198]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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.
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

#### 5.40 CALCULATE ROM CHECKSUM [197]

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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.
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

### 5.41 REQUEST CREATION DATE [196]

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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 Table 10:

**Table 10: 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 REQUEST BASE YEAR [170]. 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.

b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

### **5.42 REQUEST LAST MODIFICATION DATE [195]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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 REQUEST CREATION DATE [196].

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.

b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

### **5.43 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 REQUEST INSERTION COUNTER [226].

Command protected by a PIN.

Notes:

- a) A NACK is returned if the Exit module does not have its software correctly loaded
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

#### **5.44 REQUEST FRAUD COUNTER [193]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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 REQUEST INSERTION COUNTER [226].

Command protected by a PIN.

Notes:

- a) A NACK is returned if the Exit module does not have its software correctly loaded
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

#### **5.45 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 REQUEST PRODUCT CODE [244], 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'] [chk]

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

## 5.46 MODIFY DEFAULT SORTER PATH [189]

This command is not available if the Validator is programmed in "L66 compatibility" mode.

The machine communicates to the Validator the default classification path to which the coins are sent.

Send: [Dir] [1] [1] [189] [Data 1] [Chk]

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

Where Dir = address of the corresponding validator

Data 1 = The default path of the sorter that the validator should send the coins.



The value of the default classification path is stored in the Flash memory of the Validator.

The values that [Data 1] should take are one of the following:

- 1: path 1
- 2: path 2
- 3: path 3
- 4: path 4
- 5: path 5

The value is initially set to path 1.

Command protected by a PIN.

Notes:

a) A NACK is returned if:

- The Exit module does not have its software correctly loaded
- The classification path has an incorrect value: it is not any of the values previously described.
- There has been an error in programming the value of the default path in the Flash memory of the Validator.

b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

**Example:**

Send: [2] [1] [1] [189] [3] [60]

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

The validator should send the coins to path 3 by default.

## **5.47 REQUEST DEFAULT SORTER PATH [188]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.



This command requests the Validator for information about the default classification path to which the coins are sent.

Send: [Dir] [0] [1] [188] [Chk]

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

Where Dir = address of the *corresponding validator*

Data 1 = Path of the sorter that is taken as default.

The value sent corresponds to that which is programmed using the command MODIFY DEFAULT SORTER PATH [189].

Command protected by a PIN.

Notes:

- a) A NACK is returned if the Exit module does not have its software correctly loaded.
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

## 5.48 MODIFY COIN ID [185]

Command not available in "Secure Mode".

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 REQUEST COIN ID [184]. 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.

b) This command is not available if the Validator is working in "Secure Mode".

**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]

## 5.49 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 MODIFY COIN ID [185].

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

## 5.50 MODIFY SECURITY SETTING [181]

Command not available in "Secure Mode".

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.
- b) This command is not available if the Validator is working in "Secure Mode".

## **5.51 REQUEST SECURITY SETTING [180]**

Command not available in "Secure Mode".

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 MODIFY SECURITY SETTING [181].

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
- b) This command is not available if the Validator is working in "Secure Mode".

## 5.52 MODIFY BANK SELECT [179]

Command not available in "Secure Mode".

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 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.

b) This command is not available if the Validator is working in "Secure Mode".

### **5.53 REQUEST BANK SELECT [178]**

Command not available in "Secure Mode".

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 MODIFY BANK SELECT [179].

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.

b) This command is not available if the Validator is working in "Secure Mode".

### **5.54 REQUEST ALARM COUNTER [176]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.



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
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

## **5.55 REQUEST BASE YEAR [170]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.



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.

Notes:

- a) A NACK is returned if the Exit module does not have its software correctly loaded
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

## 5.56 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.

### **5.57 MODIFY INHIBIT AND OVERRIDE REGISTER [162]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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.

Notes:

- a) A NACK is returned if the Exit module does not have its software correctly loaded.
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode.

### **5.58 REQUEST FIRMWARE UPGRADE CAPABILITY [141]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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

0: Firmware not updateable if the Validator is working in "Secure Mode".

The Validator allows the remote modification of the firmware: see commands BEGIN FIRMWARE UPGRADE [139], UPLOAD FIRMWARE [140], page 81, and FINISH FIRMWARE UPGRADE [138].

Command protected by a PIN.

Notes:

a) The Validator does not answer if it is programmed in "L66 compatibility" mode

## 5.59 UPLOAD FIRMWARE [140]

Command not available in "Secure Mode".

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 4.3.

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.

Notes:

- a) A NACK is returned if:
  - The command BEGIN FIRMWARE UPGRADE [139], see 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.
- d) This command is not available if the Validator is working in "Secure Mode".

## **5.60 BEGIN FIRMWARE UPGRADE [139]**

Command not available in "Secure Mode".

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.

Notes:



- a) A NACK is returned if the Validator has not been able to start the modification of the firmware.
- b) This command is not available if the Validator is working in "Secure Mode".

### 5.61 FINISH FIRMWARE UPGRADE [138]

Command not available in "Secure Mode".

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.

Notes:

- 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.
- c) This command is not available if the Validator is working in "Secure Mode".

### 5.62 SET ACCEPTANCE LIMIT [135]

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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 5.20- READ BUFFERED CREDIT OR ERROR CODES [229].

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.

Notes:

- a) A NACK is returned if the Validator does not have its software correctly loaded.
- b) The Validator does not answer if it is programmed in "L66 compatibility" mode

### **5.63 BEGIN TABLES UPLOAD [99]**

Command not available in "Secure Mode".

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.



Note:

- 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.
- c) This command is not available if the Validator is working in "Secure Mode".

## 5.64 UPLOAD TABLES [98]

Command not available in "Secure Mode".

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 UPLOAD FIRMWARE [140].

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.

Note:

- a) A NACK is returned if:
  - The Validator does not have its software correctly loaded.
  - The command BEGIN TABLES UPLOAD [99], 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.
- c) This command is not available if the Validator is working in "Secure Mode".

### 5.65 FINISH TABLES UPLOAD [97]

Command not available in "Secure Mode".

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.

Note:

- 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.
- d) This command is not available if the Validator is working in "Secure Mode".

### 5.66 REQUEST MODULES INFORMATION [96]

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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 Table 11).
  - “41XXXXXX-X,vXX” if the coin programming in both modules are correct and 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 Table 11.

**Table 11: Command “Request Modules Information”: error codes**

<b>Code</b>	<b>Description</b>
0	No errors
1	Sensor Module not connected or it is not working
2	Incorrect firmware of the Exit Module
3	Incorrect firmware of the Sensor Module
4	Validator (Sensor Module or Exit Module) with incorrect programming or not programmed.



5	Incorrect values of the Exit Module configuration data (user data)
6	Countries id. of the coin programming not compatible
7	The programming data references of the Sensor Module and Exit Module are not coincident

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 REQUEST SERIAL NUMBER [242].

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

Notes:

- a) The Validator does not answer if it is programmed in "L66 compatibility" mode

### **5.67 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 = 3

Command protected by a PIN.

### **5.68 CLEAR COMMS STATUS VARIABLES [ 3 ]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

Reset to zero all the counters named in the command REQUEST COMMS STATUS VARIABLES [2]:

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

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

Command protected by a PIN.

Note:

a) The Validator does not answer if it is programmed in "L66 compatibility" mode

### **5.69 REQUEST COMMS STATUS VARIABLES [ 2 ]**

This command is not available if the Validator is programmed in "L66 compatibility" mode.

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\_checksum: 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 CLEAR COMMS STATUS VARIABLES [3].

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

Command protected by a PIN.

Note:

a) The Validator does not answer if it is programmed in "L66 compatibility" mode



## 5.70 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 coin counter information and the configuration of inhibit bits is lost.

Command protected by a PIN.

## 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 REQUEST SERIAL NUMBER [242].
- Information on the modules that the Validator is made up of: command REQUEST MODULES INFORMATION [96].

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 REQUEST MODULES INFORMATION [96], 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 TELE-PROGRAMMING. 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.

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.