AZKOYEN
MEDIOS DE PAGO

Modular X6 ccTalk DSP

AZKOYEN
MEDIOS DE PAGO

Contenido

1 INTRODUCTION.

This manual describes the way of working of the coin validators in the XccTalk -DSP range that are part of the Modular DSP series. The points covered in this document are therefore valid for the following models of validators:

- X6-D2S ccTalk
- X6-D4S ccTalk

This manual will use general term XccTalk-DSP, which shall refer to any of the mentioned validators.

For full information on the operation of these validators, this manual must be complemented with the corresponding manual "Technical Features" of the product available on the Azkoyen website: "http://sat.azkoyen.com".

2 WAY OF WORKING. CCTALK® SERIAL COMMUNICATION PROTOCOL.

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

2.1 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 optin (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: conNector Type 7.

2.2 PHYSICAL LEVEL

2.2.1 Voltage levels and transmission speed.

The data is established using TTL levels:

- 5 volts is a '1' logic
- 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

2.2.2 Connectors.

The validator has two possible connector to use through ccTalk protocol:

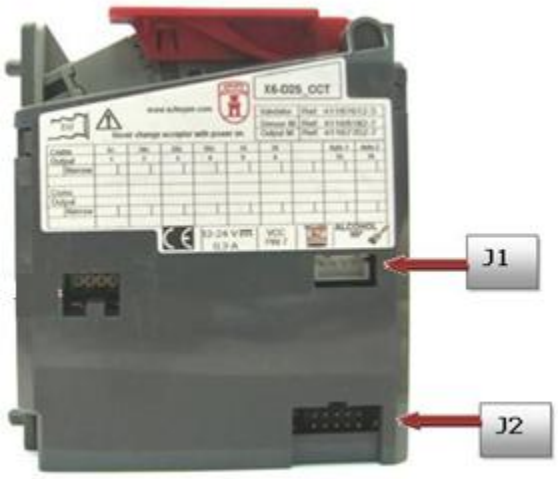
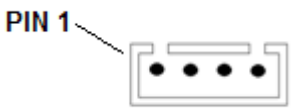


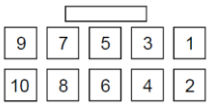
Figure 1. Connectors accessible to the user.

✓ **Connector J1 (4 pins) for ccTalk communication with the machine.**



Connector J1. CONnECTOR CCTALK		
Pins	**Function**	**Notes**
Pin 1	Vcc	12–24 Vcc
Pin 2	N.C.	Not used
Pin 3	GND	
Pin 4	/DATA	Data

✓ **Connector J2 (10 pins) for ccTalk communication with the machine.**



The function of each of the pins is shown in next table

Connector J1 on the Output module		
Pins	**Function**	**Notes**
Pin 1	DATA	DATA LINE
Pin 2	N.C.	Not used
Pin 3	S1	Activation coil 1 in sorter
Pin 4	S3	Activation coil 3 in sorter
Pin 5	N.C.	Not used
Pin 6	S2	Activation coil 2 in sorter
Pin 7	Vcc	Power supply
Pin 8	GND	
Pin 9	N.C	Not used
Pin 10	N.C	Not used

Table 1. Connector J1 (2x5)

✓ 0 Vcc to 0.7 Vcc – Logic "0"

✓ >4Vcc, or pin not connected: logic "1"

Note 1: This connector is ready to work with a sorter 5 or 3 ways.

2.3 LOGIC LEVEL

2.3.1 Message structure

The format of the messages is the following:

[Destination address]

[Nº of data bytes]

[Origin address]

[Header]

[Data 1]

[Data 2]

...

[Data N]

[Checksum]

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

2.3.2 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.

2.3.3 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 STATUS VARIABLES [2]).

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.

2.3.5 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.

2.3.6 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.

2.3.7 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'.

Example :
For message [01][00][02][00] the checksum will be [253], because:

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

For message[02][01][01][179] [02] the checksum will be [71], because:

2 + 1 + 1 + 179 + 2 + 71 = 256

2.3.8 Timing conditions.

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.

2.4 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.

3 COMMANDS LIST

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

All commands, excepting "Address Poll (253)" can be protected by PIN.

If it receives a command and a valid a PIN has not previously been introduced, the Validator will not answer the command. The value a PIN = 0 will deactivate this protection.

In next table it is possible to find a list of the ccTalk commands implemented in X6ccTalk validator series. More detailed information can be found in following paragraphs.

Cabecera (decimal)	Comando
FEh	SIMPLE POLL [254]
FDh	ADDRESS POLL [253]
FCh	ADDRESS CLASH [252]
FBh	ADDRESS CHANGE [251]
FAh	ADDRESS RANDOM [250]
F9h	REQUEST POLLING PRIORITY [249]
F8h	REQUEST STATUS [248]
F6h	REQUEST MANUFACTURER ID [246]
F5h	REQUEST EQUIPMENT CATEGORY ID [245]
F4h	Command protected by a PIN. REQUEST PRODUCT CODE [244]
F3h	REQUEST DATABASE VERSION [243]
F2h	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.* REQUEST SERIAL NUMBER [242]
F1h	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.* REQUEST SOFTWARE REVISION [241]
F0h	TEST SOLENOIDS [240]
ECh	READ OPTO STATES [236]
E8h	PERFORM SELF-CHECK [232]
E7h	MODIFY INHIBIT STATUS [231]
E6h	REQUEST INHIBIT STATUS [230]
E5h	READ BUFFERED CREDIT OR ERROR CODES [229]
E4h	MODIFY MASTER INHIBIT STATUS [228]
E3h	A NACK is returned if the *Exit module does not have its software correctly loaded* REQUEST MASTER INHIBIT STATUS [227]
E2h	A NACK is returned if *the Exit module does not have its software correctly loaded* *i. This bit will be at 0: normal working mode, after a reset or power-up.* REQUEST INSERTION COUNTER [226]
E1h	REQUEST ACCEPT COUNTER [225]
DEh	A NACK is returned if the *Exit module does not have its software correctly loaded* MODIFY SORTER OVERRIDE STATUS [222]
DDh	REQUEST SORTER OVERRIDE STATUS [221]
DBh	ENTER NEW PIN NUMBER [219]
DAh	ENTER PIN NUMBER [218]
D8h	This command always responds ACK, *independently of if the PIN introduced is correct or not.* REQUEST DATA STORAGE AVAILABILITY [216]
D7h	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:* *i. A NACK is returned if the Exit module does*

	not have its software correctly loaded READ DATA BLOCK [215]
D6h	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.* WRITE DATA BLOCK [214]
D5h	REQUEST OPTION FLAGS [213]Request option flags
D4h	REQUEST COIN POSITION [212]
D2h	MODIFY SORTER PATHS [210]
D1h	REQUEST SORTER PATHS [209]
CAh	TEACH MODE CONTROL [202]
C9h	REQUEST TEACH STATUS [201]
C7h	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.* *ii. BUSY is returned (6)* ▪ *if the Validator is busy measuring or accepting a coin at the moment of receiving the command.* CONFIGURATION TO EEPROM [199]
C6h	COUNTERS TO EEPROM [198]
C5h	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.* CALCULATE ROM CHECKSUM [197]
C4h	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.*

	REQUEST CREATION DATE [196]
C3h	A NACK is returned if - *The Exit module does not have its software correctly loaded.* - *The Validator does not have its tables correctly programmed.* REQUEST LAST MODIFICATION DATE [195]
C2h	A NACK is returned if ▪ *The Exit module does not have its software correctly loaded.* ▪ *The Validator does not have its tables correctly programmed.* REQUEST REJECT COUNTER [194]
C1h	A NACK is returned if *the Exit module does not have its software correctly loaded* REQUEST FRAUD COUNTER [193]
C0h	A NACK is returned if *the Exit module does not have its software correctly loaded* REQUEST BUILD CODE [192]
BDh	MMODIFY DEFAULT SORTER PATH [189]
BCh	REQUEST DEFAULT SORTER PATH [188]
B9h	the Exit module does not have its software correctly loaded. MODIFY COIN ID [185]
B8h	REQUEST COIN ID [184]
B5h	MODIFY SECURITY SETTING [181]
B4h	REQUEST SECURITY SETTING [180]
B3h	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* MODIFY BANK SELECT [179]
B2h	REQUEST BANK SELECT [178]
B0h	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.*

	REQUEST ALARM COUNTER [176]
AAh	A NACK is returned if *the Exit module does not have its software correctly loaded* REQUEST BASE YEAR [170]
A9h	A NACK is returned if *the Exit module does not have its software correctly loaded* REQUEST ADDRESS MODE [169]
A2h	Command protected by a PIN. MODIFY INHIBIT AND OVERRIDE REGISTER [162]
8Dh	RREQUEST FIRMWARE UPGRADE CAPABILITY [141]
8Ch	UPLOAD FIRMWARE [140]
8Bh	BEGIN FIRMWARE UPGRADE [139]
8Ah	A NACK is returned if *the Validator has not been able to start the modification of the firmware.* FINISH FIRMWARE UPGRADE [138]
87h	A NACK is returned if *the Validator has not been able to finish the modification of the firmware.* *i. It may be necessary to modify the programming of coins of the Validator after transmitting the new firmware to the Validator.* SET ACCEPTANCE LIMIT [135]
63h	A NACK is returned if *the Validator does not have its software correctly loaded.* BEGIN TABLES UPLOAD [99]
62h	UPLOAD TABLES [98]
61h	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.* *ii. After the coin tables of the Validator have been modified, any coin introduced will be rejected.* FINISH TABLES UPLOAD [97]
60h	This command indicates to the Validator that the modification of the coin tables has finished.

	Command protected by a PIN. Note: *i. 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.* REQUEST MODULES INFORMATION [96]
4h	REQUEST COMMS REVISION [4]
3h	CLEAR COMMS STATUS VARIABLES [3]
2h	REQUEST COMMS STATUS VARIABLES [2]
1h	RESET DEVICE [1]

Tabla 1. Listado de commandos.

3.1 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]
Replay:	[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.

Command protected by a PIN.

3.2 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]
Replay:	{Delay variable} [Dir]

Dir=address of Selector

The algorithm to calculate {Delay variable} is the following one:

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: Notas:

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

ii. This is the only command not protected by PIN.

3.3 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 that 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
Replay:	{delay variable} [Dir]

Dir=address of validator.

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

R = rand (256)

Deactivate port Rx

Delay (4 x R) ms

Send [Dir]

Delay 1200 – (4 x R) ms

Activate port Rx

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

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

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

Notes:

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

3.4 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]	Data 1=new address for validator
Replay:	[1] [0] [Dir] [0] [Chk]	ACK without Data
		Dir=current validator address

Command protected by a PIN.

Notes:

i. *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].*

ii. *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]*

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

Example:

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

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

Replay: [1] [0] [3] [0] [252]

After the reply the Validator is programmed with address 13

3.5 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]	Dir=validator address
Replay:	[1] [0] [Dir] [0] [Chk]	ACK without Data

Command protected by a PIN.

Notes:

i. *The reply ACK is carry out with the original address original [Dir].*

ii. *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.*

iii. *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] Respuesta: [1] [0] [3] [0] [252] After the reply the Validator is programmed with a random address

3.6 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]	Dir=Validator address
Replay: [1][2][Dir][0][Data1=2][Dato2=20][Chk]	Data1:units of time used=2(units of 10 mili secs)
	Data2:value in time units=20(20 units of 10 msecs =200mili secs)

Command protected by a PIN.

3.7 REQUEST STATUS [248]

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

Send: [Dir] [0] [1] [248] [Chk]	Dir=address of validator.
Replay: [1] [1] [Dir] [0] [Dato 1] [Chk]	Dato 1: Status according to table 4.

Código	Error
0	O.K.
1	Coin return mechanism activated
2	String detector system activated

Tabla 2 Código de Status

Command protected by a PIN.

Notes:

i. *This command will not carry out a complete check of the validator. For this use the command PERFORM SELF-CHECK [232], is recommended.*

3.8 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]

Replay: [1][3][Dir][0][Data1][Data2][Data3][Chk]

Dir= address of validator.

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:

i. *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.*

3.9 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]	Dir= address of validator.
Replay:	[1][13][Dir][0]['C']['o']['i']['n'][' ']['A']['c']['c']['e']['p']['t']['o']['r'][Chk]	

Command protected by a PIN.

3.10 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]	Dir= address of validator. Data1…DataN= ASCII DATA n = Number of data to be sent
Replay:	[1][n][Dir][0][Data1]..[Data N][Chk]	

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.

3.11 REQUEST DATABASE VERSION [243]

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

Send:	[Dir] [0] [1] [243] [Chk]	Dir= address of validator. Data1=1-255: version of tables in sensor module
Replay:	[1] [1] [Dir] [0] [Data 1] [Chk]	

Command protected by a PIN.

Notes:

i. 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.*

3.12 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] Replay: [1][3][Dir][0][LSB][2SB][MSB][Chk]]	Dir= address of validator. LSB,2SB,MSB=bytes in serial number

Command protected by a PIN.

Notes:

i. 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.*

3.13 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] Replay: [1][n][Dir][0][Data 1]...[Data N][Chk]	Dir= address of validator. Data1-DataN=characters ASCII n =Number of data to be sent

The data to send consists of:

- **Product code**: same string that the command REQUEST PRODUCTO CODE [224] 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] Replay: [1] [14] [2] [0] ['X'] ['6'] [' '] ['c'] ['c'] ['t'] ['a'] ['l'] ['k'] [' '] ['V'] ['1'] ['.'] ['2'] [chk]

3.14 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]	Dir= address of validator.
Replay:	[1] [0] [Dir] [0] [Chk]	ACK without Data

[Data 1] = Bit mask of solenoid activation

Bit 0 – Acceptance gaate

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:

i. *The Exit module does not have its software correctly loaded it returns a NACK.*
ii. *There is a coin inside or the Validator is activating the exits or the solenoids at the moment of reception of the command.*
iii. *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] Replay: [1] [0] [2] [0] [253] después de 1 segundo (2 x 500 mseg.) Note: Check visually that the path of acceptance opens and it moves the sorter.

3.15 READ OPTO STATES [236]

The validator returns the switches status and the string detector mechanism.

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

Dir= address of validator.

Dato1 =Bits mask

Bits 0,4,5,6 y 7 Not used.

Bit1: 0 all the optos are free

1 any is covered.

Bit 2: Fototransistor output

0 (free) 1 (covered)

Bit 3: String detector mechanism (0 = closed / 1 = open)

Command protected by a PIN.

Notes:

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

Example
Machine asks validator 2 for its optos status Send: [2] [0] [1] [236] [17] Replay: [1] [1] [2] [0] [12] [249] Note: According to its replay (12 = 0x0C) the antifishing is opened and opto sensor is blocked.

3.16 PERFORM SELF-CHECK [232]

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

Send: [Dir] [0] [1] [232] [Chk] Replay: [1][1/2][Dir][0][Dato 1] / [Dato 2][Chk]	Dir= address of validator. Dato1/Dato2=códigos error. Ver tabla.

Code	Type of Error [Data 1]	Complementary Information [Data 2]
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	-

Tabla 3 Test de Hardware - Códigos de Error

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.

Notas:

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

ii. 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] Replay: [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] Replay:[1] [1] [2] [0] [0] [252] -> Selector OK

3.17 MODIFY INHIBIT STATUS [231]

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

Send: [Dir][2][2][231][Dato 1][Dato 2][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= address of validator. Dato1: Byte 1 inhibition (LSB), coins 1 to 8 Dato2: Byte 2 inhibition (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:

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

ii. *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].*

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] Replay: [1] [0] [2] [0] [253]

3.18 REQUEST INHIBIT STATUS [230]

This command requests the Validator for information about the configuration of the inhibit bits of the coins. This command is complementary to MODIFY INHIBIT STATUS [231].

Send:	[Dir] [0] [1] [230] [Chk]
Replay:	[1] [2] [Dir] [0] [Data 1] [Data 2] [Chk]

Dir= address of validator.
Data1: Byte 1 inhibition (LSB), coins 1 to 8
Data2: Byte 2 inhibition (MSB), coins 9 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:

i. 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 inhibit bits.

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

Replay: [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

3.19 READ BUFFERED CREDIT OR ERROR CODES [229]

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

Send:	[Dir] [0] [1] [229] [Chk]
Replay:	[1][11][Dir][0][Data1][Data2]...Data11][Chk]

Dir= address of validator.

- Data 1=Event counter:
  - ✓ 0–switch ON or reset,
  - ✓ 1 to 255–counter.

- Data 2=resul1A:
  - ✓ 0–Error,
  - ✓ 1 a 16–credit (outupt code for accepted coin)
- Data 3=result1B:
  - ✓ 1 a 255–error,
  - ✓ 1 a 5–Sorter way.

.....

- Data 10=result5A
- Data 11=result5B

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

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

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

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

- result(x)A:
  - ✓ indicates that a error has occurred (value = 0)
  - ✓ code of the coin that has been introduced (value = 1 to 16)
- result(x)B:
  - ✓ **error code** (if result(x)A = 0). See TABLE.
  - ✓ **Sorter way** for coin (if result(x)A is not 0)

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

Code	Error
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 PRIORIITY [249]. On receiving command 229 the timer is reset.

Command protected by a PIN.

Notes:

i. *A NACK is returned if the Exit module does not have its software correctly loaded*
ii. *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]

Replay:[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.

3.20 MODIFY MASTER INHIBIT STATUS [228]

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

Send:	[Dir] [1] [2] [228] [Dato 1] [Chk]
Replay:	[1] [0] [Dir] [0] [Chk]

Dir= address of validator.

Data1.bit0:=0: inhibiton activated, No coins accepted

Data1.bit0=1:Normal way of working

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

Command protected by a PIN.

Notes:

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

3.21 REQUEST MASTER INHIBIT STATUS [227]

Request information from the validator about of the present state of the "Master Inhibit":

Send:	[Dir] [0] [2] [227] [Chk]
Replay:	[1] [1] [Dir] [0] [Dato 1][Chk]

Dir= address of validator.

Data1.bit0:=0: Inhibition activated, No coins accepted

Data1.bit0=1: normal way of working

This bit is modified using the command MODIFY MASTER INHIBIT STATUS [228].

Command protected by a PIN.

Notes:

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

iii. This bit will be at 0: normal working mode, after a reset or power-up.

3.22 REQUEST INSERTION COUNTER [226]

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

Send:	[Dir] [0] [1] [226] [Chk]
Replay:	[1][3][Dir][0][Dat1][Dat2][Dat3][Chk]

Dir= address of validator.

Dat1=nº Coins insertaded (LSB)

Dat2=nº Coins insertaded (2SB)

Dat3=nº Coins insertaded (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 EEPROM [198], 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:

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

Example:
The machine asks to validatora about the counter for inserted coins. La Máquina solicita al Selector número 2 información sobre el contador de monedas insertadas. Envío: [2] [0] [1] [226] [27] Respuesta: [1] [3] [2] [0] [208] [57] [2] [197] According to its answer, there were 145.872 coins inserted: LSB = 208, 2SB = 57, MSB = 2. Now command COUNTERS TO EEPROM [198] is used: Send: [2] [0] [1] [198] [55] Replay:[1] [0] [2] [0] [253] Validator stores the counters into the Flash memory. After a power OFF and On, the validator will be able to get this value.

3.23 REQUEST ACCEPT COUNTER [225]

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

Send: [Dir] [0] [1] [225] [Chk]
Replay: [1][3][Dir][0][Dat1][Dat2][Dat3][Chk]

Dir= address of validator.

Dat1=nº Coins insertaded (LSB)

Dat2= nº Coins insertaded (2SB)

Dat3= nº Coins insertaded (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:

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

3.24 MODIFY SORTER OVERRIDE STATUS [222]

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] Replay: [1] [0] [Dir] [0] [Chk]	Dir= address of validator. Data1=bits que for allowed channels Data1.bit0=Channel 1. Value=1: allowed Data1.bit1= Channel 2. Value =1: allowed …… Data1.bit7= Channel 8. Value =1: allowed

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 OVERRRIDE STATUS [221].

Command protected by a PIN.

Notes:

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

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] Replay:[1] [0] [2] [0] [253]

3.25 REQUEST SORTER OVERRIDE STATUS [221]

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

Send: [Dir][0][1][221][Chk]
Replay: [1][1][Dir][0][Data1][[Chk]

Dir= address of validator.

Dato1=bits for allowed channels

Data1.bit0= Channel 1. Value=1: allowed

Data1.bit1= Channel 2. Value=1: allowed

……

Data1.bit7= Channel 8. Value=1: allowed

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

Command protected by a PIN.

Notes:

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

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] Replay:[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:

3.26 ENTER NEW PIN NUMBER [219]

The Validator can be protected by a number (a PIN) so that no command can be executed (except the command ADDRESSS POLL [253]) if the correct PIN has not been previously introduced.

By default, the validator has PIN number a 0 0 0 0, which means that is deactivated.

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

Send: [Dir][4][1][219][Dat1][Dat2][Dat3][Dat4][Chk]

Replay: [1] [0] [Dir] [0] [Chk]

Dir= address of validator.
Dat1,Dat2,Dat3,Dat4= 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:

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

3.27 ENTER PIN NUMBER [218]

The Validator can be protected by a number (a PIN) so that no command can be executed (except the command ADDRESSS POLL [253]) 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.

If the PIN system is deactivated (that is, pin = 0 0 0 0), it is not needed to write any pin number.

Send: [Dir][4][1][218][Dat1][Dat2][Dat3][Dat4][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir=valiator address Dat1,Dat2,Dat3,Dat4= PIN NUMBER

Note:

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

3.28 REQUEST DATA STORAGE AVAILABILITY [216]

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

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

Replay: [1][5][Dir][0][Data1][Data2][Data3][Data4][Data5][Chk]

where Dir = address of validator

Data1 =**2**: Type of memory: non-volatile (permanent), with limited use in number of writes.

Data2 =**2**: Number of blocks of read data available

Data3=**32**: Number of bytes per block of read data.

Data4=**2**: Number of blocks of write data available

Data5=**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:

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

3.29 READ DATA BLOCK [215]

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

Send: [Dir][N][1][215][Dat1][Data2(optional)][Chk]	Dir=validator address
Replay: [1][M][Dir][0][Data 3] ... [Data (M+2)][Chk]	N =1: if Data 2 is not sent 2: if it is sent

Data 1 = **0, 1**: block number to read

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

- 1 .. 64 block 0: [Dato 1] = 0
- 1 .. 32 block 1: [Dato 1] = 1

M: number of data bytes given.

- If there are no optional arguments , [Dato M]=32 bytes.
- If the olptional argument is sent = [Dato 2].

Command protected by a PIN.

Notes:

i. 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.*

3.30 WRITE DATA BLOCK [214]

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

Send: [Dir][N][1][214][Dato1][Dato2]...[Dato M][Chk]	Dir=validator address Dato1=0/1=block number Dato 2/M=data to write
Replay: [1] [0] [Dir] [0] [Chk]	

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

I. *Read the 64 data bytes: command "Read data block".*
[Dir] [2] [1] [215][block = 0][num. of Bytes = 64] [Chk]
II. *Modify the received data with the new data.*
III. *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:

i. *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.

3.31 REQUEST OPTION FLAGS [213]

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

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

Dir= validator address
Data1=0=coins are shown as "coin possition"

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:

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

3.32 REQUEST COIN POSITION [212]

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

Send: [Dir][1][1][212][Data 1][Chk]
Replay: [1][2][Dir][0][Data 2][Data 3][Chk]

Dir= validator address

Data1=0= Coin code.

Data2:LSB inhibition word

Dato3:MSB inhibition word

The value returned (2 bytes: word) has the bit corresponding to the coin code requested activated to "1"..

Command protected by a PIN.

Notes:

A NACK is returned if:

i. The Exit module does not have its software correctly loaded.

ii. The coin code is greater than 16 (erroneous argument)

Example:
The machines asks to validator about the inhibition bit for coin code 3. Send: [2] [1] [1] [212] [3] [37] Replay:[1] [2] [2] [0] [4] [0] [247] Validator answers with inhibition bit 3 (00000100 =4)

3.33 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][Dato1][Dato2][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address Dato1=coin code. Dato2=assigned sorter channel

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

- Format (b):

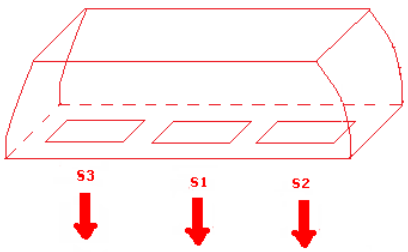
Send: [Dir][5][2][210][Data1][Data2] [Data3][Data4][Data5][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address Data1=0= Código de moneda Data2: Channel sorter for path1 Data3: Channel sorter for path2 Data4: Channel sorter for path3 Data5: Channel sorter for path4

- The priority of the classification paths is:
  - Path 1 (Data 2) (maximum priority)
  - Path 2 (Data 3) (second priority): if path 1 is not allowed.
  - Path 3 (Data 4) (third priority): if path 2 is not allowed.
  - Path 4 (Data 5) (minimum priority): if path 3 is not allowed.
- If path 4 is prohibited the coin is sent to the default path.

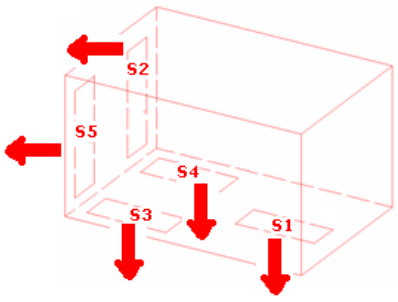
In both formats, channel value=0 means default channel

Values mean::

- 0: Default Channel:S1
- 1: Channel S1
- 2: Channel S2
- 3: Channel S3
- 4: Channel S4
- 5: Channel S5



Lineal sorter.



Modular sorter 3/5 ways

Figura 1. Clasificador.

To permit or prohibit a determined path the command MOFIFY 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:

i. 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.*

Example Format (a):
The machines configures validator 2 to exit coin number 6 by path1 (channel D). If it is busy, it will go by default path. Send : [2] [2] [1] [210] [6] [1] [34] Replay:[1] [0] [2] [0] [253]
Example Format (b)
The machines configures validator 2 to exit coin number 6 by path1 (channel D). If it is busy, will should out by channel 2 (cannel C) and if it is busy, by channel 3 (D) …. Send: [2] [5] [1] [210] [6] [1] [2] [3] [0] [26] Replay:[1] [0] [2] [0] [253]

3.34 REQUEST SORTER PATHS [209]

This command requests the Validator for information about the assigned sorter path of a coin.

Send:	[Dir][1][1][209][Data 1][Chk]
Replay:	:[1][5][Dir][0][Data2][Data3][Data4][Data5][Chk]

Dir= validator address

Data1=0= Coin code.

Data2:Sorter way for path 1

Data3: Sorter way for path 2

Data4: Sorter way for path 3

Data5: Sorter way for path 4

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

Command protected by a PIN.

Notes:

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

Example
The machines asks to validator about information for the sorter channel assigned to coin code 6. Send: [2] [1] [1] [209] [6] [36] Replay: [1] [1] [2] [0] [4] [3] [2] [1] [242] Here we can see that coin 6 goes to sorter path 4. If it is busy, will go to channel 3, 2 and 1 respectively..

3.35 TEACH MODE CONTROL [202]

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

:

Send: [Dir] 1][1][202][Data 1][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address Data1=possition in table that will be programmed.

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

- The codes of the auto programmable coins must be preprogrammed at factory.

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

The state of the auto programming of coins can be consulted using the command REQUEST TEACH STATUS [201].

Command protected by a PIN.

Notes:

iii. 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.*

iv. BUSY is returned (6)

- *if the Validator is busy measuring or accepting a coin at the moment of receiving the command.*

3.36 REQUEST TEACH STATUS [201]

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

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

Send: [Dir][1][1][201][Data1][Chk] Replay: [1][2][Dir][0][Data2][Data3][Chk]	Dir= validator address Data1=**0**:machine asks about autoprogramming status. =**1**:Machine is stopping the autoprogramming status.

Data2=inserted coins.

Data3=status code according to table:

Code	Error
252	Autoprogramation cancel
253	Error with Autoprogramation
254	Autoprogramation in progress
255	Autoprogramation finished

Tabla 4. Estado de la Autoprogramación

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.

Notas:

v. 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.*

vi. BUSY is returned (6)

- *if the Validator is busy measuring or accepting a coin at the moment of receiving the command.*

3.37 CONFIGURATION TO EEPROM [199]

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]

Replay: [1] [0] [Dir] [0] [Chk]

Dir= validator address

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:

i. 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.*

3.38 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] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address

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:

ii. 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.*

3.39 CALCULATE ROM CHECKSUM [197]

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

Send: [Dir] [0] [1] [197] [Chk] Replay: [1][4][Dir][0][Data1][Data2][Data3][Data4][Chk]	Dir= validator address Data1-Data4=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:

i. *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.*

3.40 REQUEST CREATION DATE [196]

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

Send: [Dir] [0] [1] [196] [Chk] Replay: [1][2][Dir][0][Data1][Data2][Chk]	Dir= validator address Data1=LSB Date/Data2=MSB Date

The format of the date is indicated in TABLE:

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

Tabla 5. Formato de fechas

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:

i. A NACK is returned if

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

3.41 REQUEST LAST MODIFICATION DATE [195]

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

Send: [Dir] [0] [1] [195] [Chk]	Dir= validator address
Replay: [1][2][Dir][0][Data1][Data2][Chk]	Data1=LSB Date/Data2=MSB Date

Same format than command 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.*

REQUEST CREATION DATE [196].

Command protected by a PIN.

Notes:

ii. A NACK is returned if

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

3.42 REQUEST REJECT COUNTER [194]

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

Send: [Dir] [0] [1] [194] [Chk]	Dir= validator address
Replay: [1][3][Dir][0][Data1][Data2][Data3][Chk]	Data1=Number of rejected coins (LSB) Data2=Number of rejected coins (2SB) Dato3=Number of rejected coins(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].

Notes:

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

3.43 REQUEST FRAUD COUNTER [193]

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

.

Send: Dir] [0] [1] [193] [Chk] Replay: [1][3][Dir][0][Data1][Data2][Data3][Chk]	Dir= validator address Data1=Number of rejected coin (LSB) Data2=Number of rejected coin (2SB) Data3=Number of rejected coin (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].

Notes:

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

3.44 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] Replay: [1][n][Dir][0][Data1]...[Data10][Chk]	Dir= validator address n = 10: Número de datos a enviar

Dato1-Dato10=cadena de caracteres

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:

i. 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 asks the validator its assembly code, which is"41123121-0" Send: [2] [0] [1] [192] [613] Reply: [01] [10] [2] [00] ['4'] ['1'] ['1'] ['2'] ['3'] ['1'] ['2'] ['1'] ['-'] ['0'] [02]

3.45 MODIFY DEFAULT SORTER PATH [189]

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

Send: [Dir] [1] [1] [189] [Data 1] [Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address Data1=Channel that sorter should take as per default.

X6cctalk DSP goes out of factory with a "1" value as cannel per default.

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

Command protected by a PIN.

Notes:

ii. 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.*

Example:
Send: [2] [1] [1] [189] [3] [60]
Replay: [1] [0] [2] [0] [253]
The validator should send the coins to path 3 by defaul (channel B)

3.46 REQUEST DEFAULT SORTER PATH [188]

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

Send: [Dir] [0] [1] [188] [Chk]	Dir= validator address
Replay: [1] [1] [Dir] [0] [Data 1] [Chk]	Data1=Sorter channel as per default

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

Command protected by a PIN.

Notes:

i. A NACK is returned if:

- the Exit module does not have its software correctly loaded.

3.47 MODIFY COIN ID [185]

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

Send: [Dir][7][1][185][Dato1][char1][char2] [char3][char4][char5][char6][Chk]	Dir= validator address Dato1=Coin possition (1 to 16)
Replay: [1] [0] [Dir] [0] [Chk]	char2...char6:characters ASCII for coin identifier.

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:

i. A NACK is returned if:

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

Example:

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

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

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

3.48 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]

Replay: [01][06][Dir][00][Data2][Data3][Data4] [Data5][Data6][Data7][Chk]

Dir= validator address

Data1=Channel sorter per default

Data2...Data6=characters ASCII for coin identifier.

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:

i. 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*

3.49 MODIFY SECURITY SETTING [181]

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

Send: [Dir][2][1][181][Data 1][Data 2][Chk]	Dir= validator address
Replay: [1] [0] [Dir] [0] [Chk]	Data1=Coin possition (1 to 16)
	Data2= **0**: Normal band
	=**1,2,3**: increase security.
	=**-1,-2,-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:

i. 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.*

3.50 REQUEST SECURITY SETTING [180]

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

:

Send: [Dir][1][1][180][Data 1][Chk]
Replay: [1][1][Dir][0][Data 2][Chk]

Dir= validator address

Data1= Coin possition (1 to 16)

Data 2 = **-1**: Wide band

0: normal band (by 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:

ii. 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*

3.51 MODIFY BANK SELECT [179]

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

Send: [Dir] [1] [1] [179] [Data 1] [Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address Data1=0, 1, 2, 3: bank number

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:

i. 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.*

3.52 REQUEST BANK SELECT [178]

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

Send: [Dir] [0] [1] [178] [Chk] Replay: [1] [1] [Dir] [0] [Data 1] [Chk]	Dir= validator address Data1=0, 1, 2, 3: current bank number

The bank of coins can be modified using the command MODIFY BANK SELECT [179].

Command protected by a PIN.

Notes:

i. 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.*

3.53 REQUEST ALARM COUNTER [176]

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

Send: [Dir] [0] [1] [176] [Chk] Replay: [1] [1] [Dir] [0] [Data 1] [Chk]	Dir= validator address Data1=0 – 255:counter of number of warns generated till 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:

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

3.54 REQUEST BASE YEAR [170]

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

Send: [Dir] [0] [1] [170] [Chk] Replay: [1][4][Dir][0]['2']['0']['0']['0'][Chk]	Dir= validator address

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

Comando protegido por PIN.

Notes:

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

3.55 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] Replay: [1][1][Dir][0][Dat1][Chk]	Dir= validator address Dat1.bit0=**0**:Address saved in ROM (not used) bit1=**0**: Address saved in RAM (not used) bit2=**1**: Address saved in EEPROM bit3=**0**:Choice of address by connector (not

used)

bit4=**0** Choice of address by jumpers in PCB (not used)

bit5=**0**: Choice of address by switch (not used)

bit6=**0**: address can be modified by serial commands: volátil not used)

bit7=**1**:address can be modified by serial commands (no volatile)

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

Command protected by a PIN.

3.56 MODIFY INHIBIT AND OVERRIDE REGISTER [162]

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

Send: [Dir][6][1][162][Data1][Data2][Dato3][Data4][Data5][Data6][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address Data1 and Data2:inhibit mask for current coin Data 3: allowed sorter channels for current coin Data 4 and Data 5: inhibit mask for next coin Data 6 allowed sorter channels for next coin

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

Command protected by a PIN.

Note:

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

3.57 REQUEST FIRMWARE UPGRADE CAPABILITY [141]

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

Send: [Dir] [0] [1] [141] [Chk]	Dir= validator address

Replay: [1][1][Dir][0][Data][Chk]	Data=1:Firmware in Flash memory 0:Not updateable

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

Command protected by a PIN.

3.58 UPLOAD FIRMWARE [140]

Using this command the firmware of the Validator is modified:

Send: [Dir][N][1][140][Data1][Data2][Data3]...[DataN][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address Data 1 =block counter: not used Dato 2 =line counter: not used Dato 3 ... Dato 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:

i. 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.*

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

3.59 BEGIN FIRMWARE UPGRADE [139]

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

Send: [Dir] [0] [1] [139] [Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address

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.

Notes:

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

3.60 FINISH FIRMWARE UPGRADE [138]

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

Send: [Dir] [0] [1] [138] [Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address

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

Command protected by a PIN.

Notes:

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

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

3.61 SET ACCEPTANCE LIMIT [135]

Mediante este comando se indica al selector cuantas monedas debe admitir antes de autoinhibirse:

Send: [Dir][0][1][135][Data1][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address Data1=maximum number of coins to accept (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.

Note:

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

3.62 BEGIN TABLES UPLOAD [99]

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

Send: [Dir] [0] [1] [99] [Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address

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

Command protected by a PIN.

Notes:

- *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.*

i. The following data are initilized::

- *Allowed channels: all*
- *Default channel: A*
- *Inhibitions: 0*
- *counters: 0*
- *sorter paths : all coins will go to default channel.*

3.63 UPLOAD TABLES [98]

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

Send: [Dir][n][1][98][Data1][Data2][Data3	Dir= validator address

]...[DataN][Chk] Replay: [1] [0] [Dir] [0] [Chk]	Data 1: block counter: not used Data 2 = line counter: not used Datao3 ...DataN = programmation of coins 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.

Notes:

iii. 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.*

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

3.64 FINISH TABLES UPLOAD [97]

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

Send: [Dir] [0] [1] [97] [Chk]] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address

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

Command protected by a PIN.

Note:

ii. 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.

3.65 REQUEST MODULES INFORMATION [96]

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

Send: [Dir] [0] [1] [96] [Chk] Replay: [1][51][Dir][0][Data1]...[Data51][Chk]	Dir= validator address Data1…Data51: 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 an coincident. The Validator is working normally.

Data 46: Module number connected. If the Sensor module has responded, the value returned is 2. If the Sensor module has not, it responds with the value 1.

Data 47: Firmware version of the Sensor module. It is a binary value from 1 to 255. In case of error the value can be one of the following:

- 0: The Sensor module does not respond
- 255: The firmware of the Sensor module is incomplete.

Data 48: Firmware version of the Exit module. It is a binary value from 1 to 255.

Data 49 and 50: Reserved for future use.

Data 51: Error code resulting from the comparison of the programming coin characteristics between the Sensor and Exit modules. See TABLE

CODE	DESCRIPTION
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.

3.66 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]

Replay: [1][3][Dir][0][Dat1][Dat2][Dat3][Chk]

Dir= validator address

Dato1=level of protocol

Dato 2 = Major revision

Dato 3 = Minor revision

The current version is La versión actualmente implementada es:

Level = **1**/ Major revision = **4**/ Minor revision = **3**

Command protected by a PIN.

3.67 CLEAR COMMS STATUS VARIABLES [3]

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

Send: [Dir] [0] [1] [3] [Chk] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address

Command protected by a PIN.

3.68 REQUEST COMMS STATUS VARIABLES [2]

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

Send: [Dir] [0] [1] [2] [Chk] Replay: [1][3][Dir][0][Dato1][Dato2][Dato3][Chk]	Dir= validator address Dato1= Rx_timeouts Dato2= Rx_bytes_ignored Dato3= Rx_bad_cheksum

- Rx_timeouts: this counter accumulates the number of times the validator gives a time-out in the reception of data. If the communication is carried out correctly this value should be as close to zero as possible
- Rx_bytes_ignored: if the machine sends the validator very long messages that are longer than the length of the validator buffer and they are lost, the number of lost bytes is added to this counter.
- Rx_bad_cheksum: this counter is incremented each time that a message with an erroneous checksum is received.

Each time that a reset or power-up is done, the validator counters will start at 0. Before using and evaluating the data received with this command it is advised to first execute the command 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.

3.69 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] Replay: [1] [0] [Dir] [0] [Chk]	Dir= validator address

After a Reset command ,all the information related to following counters will be lost:

- Inhibition bits configuration. See command MODIFY INHIBIT STATUS [231]
- Mater Inhibition value. See command MODIFY MASTER INHIBIT STATUS [228]
- Insert counter. See command A NACK is returned if *the Exit module does not have its software correctly loaded*

iv. *This bit will be at 0: normal working mode, after a reset or power-up.*

- REQUEST INSERTION COUNTER [226]
- Channel by default. See command A NACK is returned if the *Exit module does not have its software correctly loaded*
- MODIFY SORTER OVERRIDE STATUS [222].
- Channel paths for every coin. See command MODIFY SORTER PATHS [210]
- Rejected coins counter. See command A NACK is returned if
  - *The Exit module does not have its software correctly loaded.*
  - *The Validator does not have its tables correctly programmed.*
- REQUEST REJECT COUNTER [194]
- Frauds attempts counter. See command A NACK is returned if *the Exit module does not have its software correctly loaded*
- REQUEST FRAUD COUNTER [193]
- Alarm counter .See command 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.*
- REQUEST ALARM COUNTER [176]

- Number of coins to accept befor auto inhibit.. See command A NACK is returned if *the Validator has not been able to finish the modification of the firmware.*

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

- SET ACCEPTANCE LIMIT [135]

4 PROCEDIMIENTO DE PUESTA EN MARCHA

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 A 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 PROCESO DE TELEPROGRAMACIÓN

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:

i. *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.*

6 FIRMWARE PROGRAMMING PROCCESS.

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:

i. *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.*

ii. *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.*

iii. 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.

7 AZKOYEN TOOLS.

7.1 HEUS.

The HEUS (user tool) software has two basic applications in the management of AC-DSP validators.

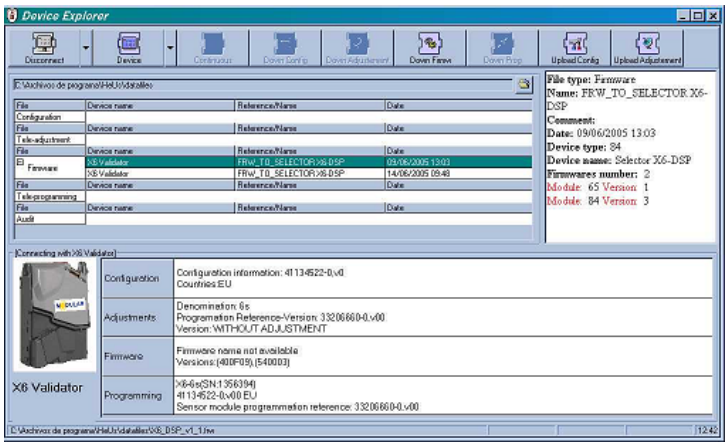


Figure 2. HeUs

- A tool for editing and modification of settings. It allows you to modify each one of the characteristic values in this range of validators detailed in paragraph 4.2.
- A tool for uploading files for configuration and programming. You can download files directly from your PC with the validator tool HEUS.

Communication between PC (HEUS) and the validator is done via cable right directly from the RS232 port of the PC to the 4-way connector J2 on the validator.

The 12/24 V power supply will not power the validator through this connector, it will always be necessary to maintain the validator powered through the J1 connector.

To know the details of the implementation and management of the HEUS, consult the specific manual available on the Azkoyen website http://sat.azkoyen.com.

7.2 TL20.

The TL20 is a hardware tool that is used to upload files for programming and configuration in AC-DSP validators.

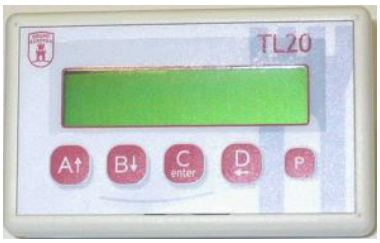


Figure 3. TL20

The TL20 programmer will connect to the validator on the 4-way connector on the validator module J2.

The 12/24 V power supply will not power the validator through this connector, it will always be necessary to maintain the validator powered through the J1 connector.

To know the details of the implementation and management of the TL20, consult the specific manual available on the Azkoyen website http://sat.azkoyen.com.

7.3 Simulation / Verification tool: IS21-ccTalk

The IS21-ccTalk interface allows us to verify the proper functioning of the validator as it simulates the behaviour of a machine.

There are numerous configuration switches as well as a Display to setup different working modes in the validator.

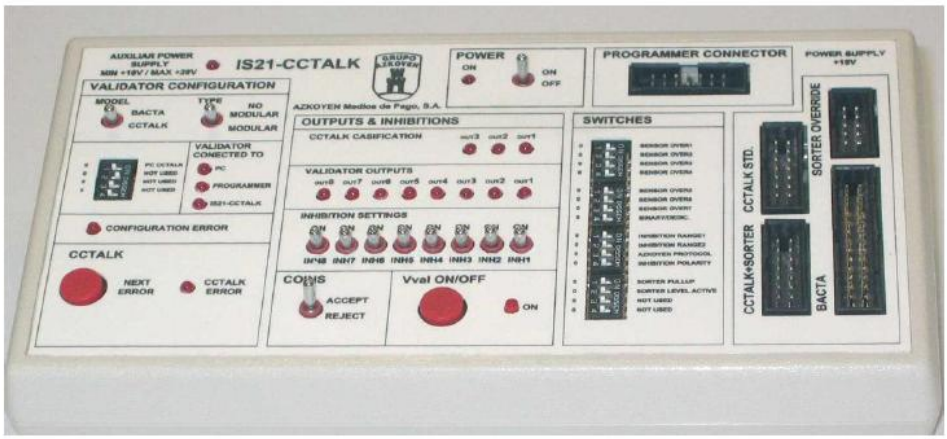
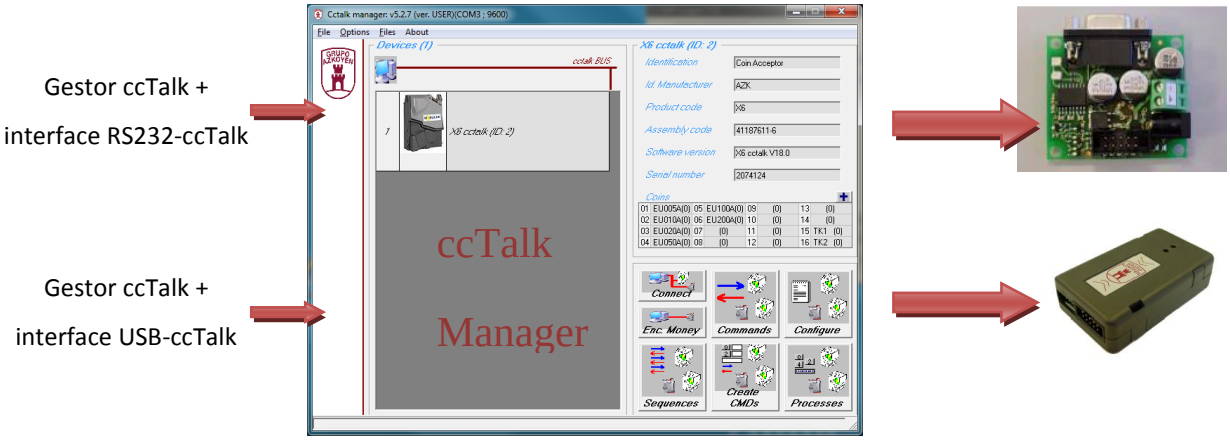


Figure 4. IS21-ccTalk

To know the details of the implementation and management of the IS21-ccTalk, consult the specific manual available on the Azkoyen website http://sat.azkoyen.com.

7.4 Simulation/Verification tools: Gestor ccTalk +Interfaces for connection.

It is possible to test the ccTalk working mode of X6ccTalk D2S, simulating the behavior of a master-slave system. In order to do it, it is used the software "ccTalk Manager" together with the "RS232-ccTalk" interface or with "USB-ccTalk" interface.



To learn more about "ccTalk Manager" software, it is advisable to read its specific manual, available in Azkoyen site http://sat.azkoyen.com.