



HOPPER – U DISCRIMINADOR



1. SERIAL COMMUNICATION PROTOCOL

The communication protocol implemented in the Hoppers in the DISCRIMINATOR range is compatible with cctalk®.

The specification used to implement this communication protocol is as follows:

cctalk Serial Communication protocol
Generic Specification Issue 4.2

cctalk® is a serial communication protocol, developed by Coin Controls Ltd. for low transmission velocity control networks.

The communication is carried out using one bidirectional line (DATA).

The use of cctalk® is open and, therefore, can be used without paying licences or rights

In the rest of this document the Payout Hoppers in the DISCRIMINATOR range will be referred to as Hopper or Hoppers

1.1. COMMUNICATION TOPOLOGY

The data is established with TTL levels, 5 volts is '0' logic and 0 volts is '1' logic which is the rest level of 5 volts

1.2. COMMUNICATION CHARACTERISTICS

The communication is asynchronous and half-duplex, that is, more than one element of the bus cannot be transmitted at the same time. The 'timing' of the communication satisfies the characteristics of the industrial standard RS232.

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

9600 bauds, 1 start bit, 8 data bits, no parity, and 1 stop bit



1.3 MESSAGE STRUCTURE

The message format is as follows:

[Checksum] [Destination address]
[No. Data bytes]
[Source address]
[Header]
[Data 1]
[Data 2]
...
[Data N]
[Checksum]

Each communication sequence is comprised of two strings.

The first corresponds to the command sent by the Machine to the Hopper and the second is the reply sent by the Hopper to the Machine. Both parts have the format indicated above.

1.3.1. Destination address

The range of addresses goes from 0 to 255 (of which 254 correspond to Hopper addresses as explained below).

- 0:** Used in messages that affect all the Hoppers simultaneously, although in the present version there is no command implemented that uses it.
- 1:** Address of the Machine
- 2:** Address of the Hopper in communication with one single slave.
- 3 to 255:** Address of the Hoppers in multi-slave communication.

The 'Destination address' byte indicates the bus node (slave) the message is addressed to.

In our case the values that are used for the Destination address are:

- 1:** When the message is sent from a Hopper to the Machine.
- 3, 4, 5 or 6:** When the message is sent to the Hopper configured by means of the dipswitch as 1, 2, 3 or 4 respectively.
- 3 to 255:** When the message is sent to the Hopper configured by means of software with MDCES commands, and with its address saved in the EEPROM.

Using MDCES commands (Multi Drop Command Extension Set), the Machine can make the Hopper have a different address than that configured with the dipswitches.



1.3.2. Number of data bytes

The range of data to be transmitted goes from 0 to 252.

This byte indicates the number of data bytes of the message and not the total number of bytes of the message. If it is '0' this means that the message has no data and in this case the total number of bytes of the message will be 5 bytes (minimum permitted).

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

1.3.3. Source address

The range of addresses goes from 1 to 255.

- 1:** Machine address.
- 2:** Address of the Hopper in communication with one single slave.
- 3 to 255:** Addresses of the Hoppers in multi-slave communication, which is the case that concerns us.

The 'Source Address' byte indicates the bus node that sends the message. In our case, the values that are used for the 'source address' are:

- 1:** When the message is sent by the machine
- 3, 4, 5 or 6:** When the message is sent by the Hopper configured with the dipswitch as 1, 2, 3 or 4, respectively.

1.3.4. Header

The header byte range goes from 0 to 255.

The Header will never have '0' value in the case of messages sent by the Machine.

In the case of reply messages, that is, those sent by the Hoppers, the Header will have '0' value in all the messages, except in the 'Negative Acknowledgement' or NACK messages.

1.3.5. Data

The range of values that each one of the data bytes can take is 0 to 255.

It has no restrictions of use, any data format being possible such as binary, ASCII, etc.

1.3.6. Checksum

Checksum is what makes the 8 less important bits of the sum of all the bytes of the message, including the actual checksum; which gives '0' as the result.

For example:

The message [01] [00] [02] [00] will be followed by checksum [253] because 1 + 0



+ 2 + 0 + 253 = 256 = 0

1.4. TIME REQUIREMENTS

1.4.1. Maximum time between bytes

The maximum time between two bytes of the same message is 50 ms. If this time is exceeded the communication program will reset the communication variables and will prepare to receive a new message.

1.4.2. Maximum time between command and reply

The maximum time to reply to a command depends on the time that it takes the Hopper to process that command.

The default time is 50 ms, if no other value is specified in the command definition.

1.4.3. Minimum time from power on until the first command is sent

The minimum time that must elapse from the time the Hopper is powered up until the first command is sent must be 250 ms.

1.5. ERROR MANAGEMENT

1.5.1. Actions when there is an error

If a Hopper receives an incomplete message (reception timeout) or with an incorrect checksum, the only action carried out is to reset the communication variables and prepare to receive a new message. The Machine, on not receiving a reply to the 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, faced with an error in the reception of a message, there is no negative acknowledgement message defined, which simplifies the implementation of multi-Hopper protocols and reduces collisions.

If a Hopper receives a command that it is not able to execute, it responds with a negative acknowledgement message. This occurs, for example, when during the execution of a payment command another payment command is received.

1.5.2. Communication of errors

The Discriminator Hopper carries out a series of checks on its peripherals and is able to detect



a series of anomalies, which are in turn notified to the Machine

The Machine is notified of the errors by means of the "Request for Status" command from the machine to the Hopper. On this request, the Hopper answers with a byte indicating its error situation and with another byte that indicates the error produced. Each one of the bits of this last byte represents a possible error, so that the bits that are on 1 will indicate that the corresponding error has been detected and the ones that are on 0 will indicate that they have not been detected.

The machine can also be notified of the errors produced through the specific cctalk® command of **[Test Hopper]** (specific to cctalk®) requested by the machine from the hopper. On this request, the Hopper answers with one byte or status register that indicates the hopper's error situation. Each one of the bits of this byte represents a possible error or operation status, so that the bits that are on 1 will indicate that the corresponding error has been detected and those that are on 0 will indicate that they have not been detected.

The errors are cumulative, that is, if more than one error is being detected at the same time, the Machine will be notified of all of them.

The Hopper quits the Error status when it receives a 'Pay' or 'Empty' or 'Reset' command, going on to execute this command. If the fault persists, the Hopper will enter the Error status again.

1.6. COMMANDS

Within the commands implemented, there are some that are specific to cctalk® and there are others that are particular to the Hopper. Of the specific cctalk® commands, there is a first group which is general, that is, valid commands for any type of device. The other specific cctalk® commands are designed to be used with Hoppers. The Discriminator Hopper can logically work as a normal Hopper, if loaded with one type of coin. In this operating mode, it may be more useful to use these commands instead of the particular commands of the Hopper.

The commands implemented are shown in the following table:



Code	Command	Data bytes SEND	Data bytes REPLY
Typical commands of cctalk®. General Commands			
FEH	Simple Poll	0	0
F5H	Request ID	0	6
F4H	Request product code	0	N
F2H	Request Serial Number	0	3
DBH	New Pin Number	4	0
DAH	Pin Number	4	0
C0H	Request built code	0	N
F6H	Request manufacturer ID	0	7
F2H	Request Serial Number	0	3
F1H	Request Software Revision	0	2
ECH	Read opto states	0	1
04H	Request Comms Revision	0	3
01H	Reset Device	0	1
Typical commands of cctalk®. Commands for Hoppers			
A4H	Enable hopper	1	1
A7H	Dispense hopper coins	1	1
A6H	Request hopper status	0	4
A3H	Test hopper	0	1
ACH	Emergency stop	0	1
A8H	Request hopper dispense count	0	3
Particular commands of the Hopper			
10H	About...	0	5
13H	Request for status	0	From 1 to 9 (See Table 2)
15H	Cancellation	0	0
19H	Multiple emptying	0	0
20H	Multiple payment	2	0
23H	Last command status	0	From 1 to 9
25H	Diameter programming	4	0
29H	Request for diameters	0	4
31H	Programming value		
32H	Value request		
35H	Intelligent payout		
13h	State of intelligent payout during execution		
23H	State of intelligent payout after execution		



1.6.1. Specific cctalk® commands. General commands

The commands described in this section are basic commands of cctalk® valid for all types of devices (not just DISCRIMINATOR hoppers) and some of them are also compulsory.

1.6.1.1. Simple poll [FEH = 254D]

Command to check the correct operation of the communication and to confirm the presence in the bus of a specific Hopper.

Send: **[Dir] [00] [01] [FEH] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk]**

If no reply is received to the request sent (Reception timeout in Machine), it will indicate that the respective Hopper is faulty or not connected.

All the cctalk® peripherals must respond to a 'Simple Poll', regardless of the cctalk® communication protocol level that has been implemented.

Example:

The machine verifies the presence of the Hopper coded as number 2 on the bus.

Send: **[04] [00] [01] [FEH] [FDH]**

Reply: **[01] [00] [04] [00] [FBH]**

The machine verifies the NON-presence of the Hopper coded as number 3 in this bus.

Send: **[05] [00] [01] [FEH] [FCH]**

Reply: **(as it is not present, there is no reply)**

1.6.1.2. Request equipment category ID [F5H = 255D]

This command permits reception from the Hopper of the chain of characters that identifies the type of device in question.

In the case of the Hopper, which is the device in question, 'Discri' is received.

Send: **[Dir] [00] [01] [F5H] [Chk]**

Reply: **[01] [06] [Dir] [00] [Data 1] [Data 2] [...] [Data 6] [Chk]**

Where: Data 1 - Data 6 = chain of characters

Dir = direction of the respective Hopper

Example:

The Machine requests Hopper number 1 the chain of characters that indicates the



type of device in question.

Send: [03] [00] [01] [F5H] [07H]

Reply: [01] [06] [03] [00] [44H] [69H] [73H] [63H] [72H] [69H] [98H]

bytes [44H] [69H] [73H] [63H] [72H] [69H] correspond to characters

'D', 'i', 's', 'c', 'r', 'i', respectively

1.6.1.3. Request product code [F4H = 244D]

With this command, the corresponding Hopper sends the machine the product code of the device. The complete identification of the product can be determined by the use of the **[Request product code]** command followed by the **[Request build code]** command.

In the case of the Hopper, which is the one that concerns us, it responds:

Send: [Dir] [00] [01] [F4H] [Chk]

Reply: [01] [n] [Dir] [00] [Data 1] [Data 2] [...] [Data n] [Chk]

Where:

Data 1 ... Data n = chain of characters

N = Amount of data to be sent

Dir = address of the corresponding Hopper.

Example:

The Machine requests hopper number 3 for its product code, which in this example is "Discri"

Send: [05] [00] [01] [F4H] [06]

Reply: [01] [06] [03] [00] [44H] [69H] [73H] [63H] [72H] [69H] [98H]

Bytes [44H] [69H] [73H] [63H] [72H] [69H] correspond to the characters 'D', 'i',

's', 'c', 'r', 'i', respectively

1.6.1.4. Request build code [F6H = 192D]

With this command, the corresponding Hopper sends the machine the assembly code of the device. The complete identification of the product can be determined by the use of the **[Request product code]** command followed by the **[Request build code]** command.

In the case of the Hopper, which is the one that concerns us, it responds "Discri":



Send: [Dir] [00] [01] [C0H] [Chk]
Reply: [01] [n] [Dir] [00] [Data 1] [Data 2] [...] [Data n] [Chk]

Where:

Data 1 ... Data n = chain of characters

N = Amount of data to be sent

Dir = address of the corresponding Hopper.

Example:

The Machine requests hopper number 3 for its assembly code, which in this example is "Discri"

Send: [05] [00] [01] [F4H] [06]
Reply: [01] [06] [03] [00] [44H] [69H] [73H] [63H] [72H] [69H] [98H]

Bytes [44H] [69H] [73H] [63H] [72H] [69H] correspond to the characters 'D', 'i', 's', 'c', 'r', 'i', respectively
--

1.6.1.5. Request manufacturer id [F6 H = 246 d]

With this command, the corresponding Hopper sends the Machine the identification of the device manufacturer.

In the case of the Hopper, which is the one that concerns us, 'Azkoyen' is received.

Send: [Dir] [00] [01] [F6H] [Chk]
Reply: [01] [07] [Dir] [00] [Data 1] [Data 2] [...] [Data 7] [Chk]

Where:

Data 1 ... Data 7 = chain of characters

Dir = address of the corresponding Hopper.

Example:

The machine requests the Hopper number 3 its manufacturer's identification code, which in this example is "Azkoyen"

Send: [05] [00] [01] [F6H] [04]
Reply: [01] [07] [03] [00] [41H] [7AH] [6BH] [6FH] [79H] [65H] [6EH]
[14H]



The bytes [41H] [7AH] [6BH] [6FH] [79H] [65H] [6EH] correspond to characters 'A', 'z', 'k', 'o', 'y', 'e' and 'n', respectively

1.6.1.6. Request software revision [F1 H = 241 d]

With this command, the corresponding Hopper sends the Machine the present software version. cctalk® does not make any restriction on the message format.

Send: **[Dir] [00] [01] [F1H] [Chk]**
Reply: **[01] [02] [Dir] [00] [Data 1] [Data 2] [Chk]**
Where: Data 1 = Byte of the version before the point
 Data 2 = Byte of the version after the point
 Dir = address of the corresponding hopper

Example:

The machine requests hopper 3 for its software version, which in this example is version 9.2.

Send: **[05] [00] [01] [F1H] [09]**
Reply: **[01] [02] [05] [00] [09] [02] [EDH]**

The hopper replies to the machine that the software version is 9.2

1.6.1.7. Read opto states [EC H = 236 d]

To this command, the Hopper responds by sending a byte, which indicates the value of the empty and full detection optos and the mechanical scales.

Send: **[Dir] [00] [01] [ECH] [Chk]**
Reply: **[01] [01] [Dir] [00] [Data 1] [Chk]**
Where:
Data 1 = value of the detectors
Data 1 .bit 0 = Empty detector
Data 1 .bit 1 = Full detector
Dir = address of the corresponding Hopper

The following table shows the value of these empty and full bits (bits 0 and 1) depending on the coin load of the Hopper:



Load	Bit 0	Bit 1
Empty	1	1
1/2 Load	0	1
Full	0	0

Example:

The Machine requests the Hopper number 2 the value of the empty, full detectors and scales.

Send: [04] [00] [01] [ECH] [0FH]
Reply: [01] [01] [04] [00] [01] [F9H]

From this reply we deduce that the Empty detector=1 and Full detector=0

1.6.1.8. Request comms revision [04H = 4D]

As a reply to this command, the Hopper sends the implementation level of the **cctalk®** protocol and the communication software version. If this version is 3.6, the 'main revision' will be 3 and the 'minor revision' will be 6.

Send: [Dir] [00] [01] [04] [Chk]
Reply: [01] [03] [Dir] [00] [Data 1] [Data 2] [Data 3] [Chk]

Where:

Data 1 = Level of the communication protocol

Data 2 = Part prior to the dot of the communication software version

Data 3 = Part following the dot of the communication software version.

Dir = address of the corresponding "Hopper.

Example:

The machine requests Hopper number 1 for information about the implementation level of the **cctalk®** protocol and the software version of the communication protocol.

Send: [03] [00] [01] [04] [F8H]
Reply: [01] [03] [03] [00] [01] [00] [00] [F8H]



From this reply it is deduced that the Hopper has implemented level 1 of the cctalk® protocol and that the software version of the communication protocol is 0.0.

1.6.1.9. Reset Device [01H = 1 d]

This command causes the Hopper to carry out a software reset. The execution of the program makes the hopper go into the reset vector.

The affected Hopper sends a positive acknowledgement string immediately before making the reset.

Example:

The Machine requests the Hopper number 1 to carry out a software reset.

Send: [03] [00] [01] [01H] [FBH]

Reply: [01] [00] [03] [00] [FCH]

1.6.1.10. Request serial number [F2 H=242 d]

In reply to this command, the Hopper sends the serial number of the device in a 3-byte code. This command has been implemented in order for the machine to include the constant safety code necessary to be able to make the coin payments.

Send: [Dir] [00] [01] [F2 H] [Chk]

Reply: [01] [03] [Dir] [00] [Serial 1 - LSB] [Serial 2] [Serial 3 - MSB] [Chk]

Example:

The Machine requests Hopper number 1 for information about the serial number, which in this case is the safety code.

Send: [03] [00] [01] [F2H] [0AH]

Reply: [01] [03] [03] [00] [4EH] [61H] [BCH] [8EH]

From this reply, it is deduced that the Hopper has the decimal number 12345678 as safety code.

1.6.1.11. Enter new PIN number [DB H= 219 d]

All the commands implemented in the hopper can be protected by a PIN.



The present PIN number can be changed with this command, but as it is a command that is also protected by a PIN, the present PIN number must first be introduced.

If a cctalk command is executed without having entered the correct PIN number (including this command), there will be no reply from the hopper.

The PIN number is a 32-bit binary code (4.294.967.296 combinations).

Modifying the PIN number to the value 0 deactivates the protection by PIN.

Send: **[Dir] [04] [01] [DB H] [PIN 1] [PIN 2] [PIN 3] [PIN 4] [Chk]**

Reply: **[01] [03] [Dir] [00] [Chk]**

1.6.2.12. Enter PIN number [DA H= 218 d]

All the commands implemented in the hopper can be protected by a PIN.

The present PIN number can be introduced with this command.

If the present PIN is different from 0, it must be entered after every power-up and reset to be able to execute any command.

If a cctalk command is executed without having entered the correct PIN number, there will be no reply from the hopper.

Whether the PIN number introduced is correct or incorrect, the hopper sends an affirmative reply.

Send: **[Dir] [04] [01] [DA H] [PIN 1] [PIN 2] [PIN 3] [PIN 4] [Chk]**

Reply: **[01] [03] [Dir] [00] [Chk]**

1.6.2. Specific commands of cctalk ®. Commands for Hoppers

This group of commands typical of cctalk® is designed to be used with coin hoppers that only contain one type of coin.

Obviously, the Hopper can be used in this way, although full use would not be made of its possibilities. If this option is chosen, it will surely be easier to use the commands described below, instead of the using the commands of the Hopper.

If the Hopper is working with two types of coins and receives a typical command of Hoppers of only one type of coin, the Hopper will carry out the corresponding action on type 1 coins (coins of smaller value).

In any case it is recommended not to use the commands for Hoppers if the Discriminator is working with two different types of coins, as if for example the Hopper has paid coins of two types in the last payment and then it is sent the **[Request Hopper**



Status] command the Discriminator will only respond about type 1 coins, not sending any information about type 2 coins.

1.6.2.1. Enable Hopper [A4H = 164 d]

This command must be used to enable the Hopper before paying coins with any payment or emptying command.

Send: **[Dir] [01] [01] [A4H] [Data 1] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk]**

Where:

Dir = address of the corresponding Hopper

Data 1 = [A5H] Hopper enabled

Other than [A5H], Hopper disabled.

Example:

The machine enables Hopper 1.

Send: **[03] [01] [01] [A4H] [A5H] [B2H]**

Reply: **[01] [00] [03] [00] [FC]**

1.6.2.2. Dispense Hopper Coins [A7H = 167 d]

This command provokes the payment of the number of type 1 coins indicated in the Data 1 byte.

If the Hopper can execute the command, it returns a positive acknowledge string and begins to make the payment until the coins indicated have been extracted, or a maximum span is detected, or the cancellation command is received or an error is detected or a hardware reset is made (power failure).

In the event that the Hopper cannot execute the command, it returns a negative acknowledgement string and continues in the status it was. Sending this negative acknowledgement may be due to the Hopper being in error or another command is being executed at that time:

Send: **[Dir] [01] [01] [A7H] [Data 1] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk]** -> Positive acknowledgement

Reply: **[01] [00] [Dir] [05] [Chk]** -> Negative acknowledgement

Where:

Dir = address of the corresponding Hopper



[Data 1] = the number of type 1 coins to pay (between 1 and 255)

Example:

The Machine requests Hopper 2 to make a payment of 89 coins of type 1.

Send: [04] [01] [01] [A7H] [4EH] [59H] [FAH]

Reply: [01] [00] [04] [00] [FB]

This command provokes the payment of the number of coins indicated in Data 4 byte, and in turns sends the necessary safety code to permit the payment to be made in Data 1, 2 and 3.

If the Hopper can execute the command, it returns a positive acknowledge string and begins to make the payment until the coins indicated have been extracted, or a maximum span is detected, or the cancellation command is received or an error is detected or a hardware reset is made (supply failure).

In the event that the Hopper cannot execute the command, it returns a negative acknowledgement string and continues in the status it was. Sending this negative acknowledgement may be due to the Hopper being in error or another command is being executed at that time, or the safety code received is not correct:

Send: **[Dir] [04] [01] [A7H] [Data 1] [Data 2] [Data 3] [Data 4] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk]** -> Positive acknowledgement

Reply: **[01] [00] [Dir] [05] [Chk]** -> Negative acknowledgement

Where:

Dir = address of the corresponding Hopper

[Data 1] = less significant byte of the safety code (LSB)

[Data 2] = Intermediate byte of the safety code

[Data 3] = Most significant byte of the safety code (MSB)

[Data 4] = Number of coins of type 1 to be paid (between 1 and 255).

Example:

The Machine requests hopper 1 to make a payment of 89 coins of type 1 with safety code 12345678.

Send: [03] [04] [01] [A7H] [4EH] [61H] [BCH] [59H] [8DH]

Reply: [01] [00] [03] [00] [FC]

This reply indicates that the Hopper has begun to execute the payment command



1.6.2.3. Request Hopper Status [A6H = 166 d]

This command requests information from the Hopper about different status parameters.

The first of them (Data 1) is the number of payments made since the last reset. When the payment number is at 255, the next payment command makes this payment counter go to value 1. The only way that this counter can be worth 0 is after a reset.

The second (Data 2), is the number of coins of type 1 that still have to be paid if the Discriminator is carrying out a payment at this time.

Parameters 3 and 4 correspond to the number of coins of type 1 paid during the last payment and the number of coins of type 1 not paid in the last payment.

Send: **[Dir] [00] [01] [A6H] [Chk]**

Reply: **[01] [04] [Dir] [00] [Data 1] [Data 2] [Data 3] [Data 4] [Chk]**

Where:

Dir = address of the corresponding Hopper

[Data 1] = Number of payments made since the last reset

[Data 2] = Number of coins of type 1 to be paid

[Data 3] = Number of coins of type 1 paid during last payment

[Data 4] = Number of coins of type 1 to be paid during last payment

Example:

The machine requests hopper 1 information about its status.

Send: [03] [00] [01] [A6H] [56]

Reply: [01] [04] [03] [00] [34H] [00] [22H] [00H] [A2H]

The Hopper has made 52 payments since the last reset, no coins are left to be paid (from where it can be deduced that it is not making any payment at this time), in the last payment it has paid 34 coins and none are left to be paid

1.6.2.4. Test Hopper [A3H = 163 d]

This command causes information to be sent about several error and operation flags of the Hopper.

Send: **[Dir] [00] [01] [A3H] [Chk]**

Reply: **[01] [01] [Dir] [00] [Data 1] [Chk]**

Where:

Dir = address of the corresponding Hopper



- Data 1 =
- Bit 0 - Maximum absolute current exceeded.
 - Bit 1 - Maximum payment time exceeded.
 - Bit 2 - Motor turned in opposite direction during last payment to eliminate a jam.
 - Bit 3 - Opto fraud attempt, release blocked during standby.
 - Bit 4 - Opto fraud attempt, short-circuit during standby.
 - Bit 5 - Opto blocked during payment.
 - Bit 6 - Hardware reset occurred
 - Bit 7 - Payment disabled.

1 = True, 0 = False.

Example:

The Machine requests hopper 1 information about several parameters.

Send: [03] [00] [01] [A3H] [59H]

Reply: [01] [01] [03] [00] [85H] [76H]

The Hopper has an excess maximum current (probably due to a jam), during the last payment the motor had to be turned in the opposite direction to undo a jam and the payment has been disabled

1.6.2.5. Emergency Stop [ACH = 172 d]

This command automatically stops the payment sequence and transmits the number of coins of type 1 that still have to be paid.

Send: [Dir] [00] [01] [ACH] [Chk]

Reply: [01] [01] [Dir] [00] [Data 1] [Chk]

Where:

Dir = address of the corresponding Hopper

Data 1 = Number of coins of type 1 that still have to be paid (between 1 and 255)

Example:

The Machine requests the Hopper to make an emergency stop.

Send: [03] [00] [01] [ACH] [50H]

Reply: [01] [01] [03] [00] [0A] [F1]



Hopper 1 has cancelled the payment command and at the time of the cancellation 10 coins of type 1 still had to be paid

1.6.2.6. Request Hopper Dispense Count [A8H = 168 d]

This command makes the hopper send the machine the number of coins paid since the last reset. The number of coins paid varies from 0 to 16.777.215.

Send: **[Dir] [00] [01] [A8H] [Chk]**
Reply: **[01] [03] [Dir] [00] [Data 1] [Data 2] [Data 3] [Chk]**
Where: Dir = address of the hopper
 [Data 1] = number of type 1 coins paid (LSB).
 [Data 2] = number of type 1 coins paid.
 [Data 3] = number of type 1 coins paid (MSB).

Example:

The machine requests hopper 1 for the number of coins paid since the last reset.

Send: **[03] [00] [01] [A8H] [54H]**
Reply: **[01] [03] [03] [00] [0A] [32H] [00] [BD]**

The number of coins paid from hopper 1 since the last reset was 12.810

1.6.3. Particular commands of the hopper

The cctalk® protocol provides a high number of commands for different peripheral devices, but, it is possible that none of the existing commands are suitable for a particular product. In cases like this, cctalk® permits using specific commands for the new product.

The headers that go from 7 to 99 (decimal) are reserved for these commands.

Below the particular commands of the Hopper that are not typical of cctalk® are described.

1.6.3.1. About the hopper [10H = 16D]

Command that requests the Hopper general information about it.

Send: **[Dir] [00] [01] [10H] [Chk]**
Reply: **[01] [04] [Dir] [00] [Data 1] [Data 2] [Data 3] [Data 4] [Data 5]
 [Chk]**

Where:



- Data 1 = Type of device (for the Hopper it will be code '02')
- Data 2 = Part prior to the dot of the software version
- Data 3 = Part after the dot of the software version
- Data 4 = Part prior to the dot of the communication soft version
- Data 5 = Part after the dot of the communication soft version
- Dir = Address of the corresponding Hopper

Example:

The Machine requests the Hopper number 2 information about the type of device in question, software version and communication protocol version.

Send [04] [00] [01] [10H] [EBH]
Reply: [01] [05] [04] [00] [02] [02] [03] [01] [04] [EFH]

From this reply it can be deduced that the Hopper number 2 is a Hopper Communication cctalk® (code 29, with version of program 2.3 and version of communication protocol 1.4

1.6.3.2. Request for status [13H = 19D]

Command that requests Hopper information about its status. If the Hopper is on 'Standby':

Send: [Dir] [00] [01] [13H] [Chk]
Reply: [01] [01] [Dir] [00] [Data 1] [Chk]

Where:

- Data 1 = 01H = Hopper on STANDBY
- Dir = address of the corresponding Hopper

If the Hopper is 'Emptying':

Send: [Dir] [00] [01] [13H] [Chk]
Reply: [01] [03] [Dir] [00] [Data 1] [Data 2] [...] [Data 5] [Chk]

Where:

- Data 1 = 19H = Hopper on multiple empty
- Data 2/3 = High/low part of the counter of coins type 1 extracted
- Data 4/5 = High/low part of the counter of coins type 2 extracted
- Dir = address of the corresponding Hopper

If the Hopper is 'Paying':



Send: **[Dir] [00] [01] [13H] [Chk]**
Reply: **[01] [05] [Dir] [00] [Data 1] [Data 2] [...] [Data 9] [Chk]**

Where:

- Data 1 = 35H = Hopper on multiple payment
- Data 2/3 = High/low part of the number of coins type 1 paid
- Data 4/5 = High/low part of the number of coins type 1 to be paid
- Data 6/7 = High/low part of the number of coins type 2 paid
- Data 8/9 = High/Low part of the number of coins type 2 to be paid
- ...
- Data 22/23 = High/low part of the number of coins type 6 paid
- Data 24/25 = High/Low part of the number of coins type 6 to be paid
- Dir = address of the corresponding Hopper

Example:

The Machine requests the Hopper number 2 information about its current status.

Send: [04] [00] [01] [13H] [E8H]

If the Hopper is on 'Standby':

Reply: [01] [01] [04] [00] [01] [FDH]

If the Hopper is 'Emptying':

Reply: [01] [03] [04] [00] [19H] [01] [43H] [00] [2AH] [66H]

This reply indicates that hopper 2 is in the process of emptying and that until that time it has extracted 323 (143H) coins type 1 and 42 (2AH) type 2

If the Hopper is 'Paying':

**Reply: [01] [03] [04] [00] [35H] [00] [0CH] [00] [18H] [00] [21H] [00]
 [0BH] [83H]**

This reply indicates that hopper 2 is in the process of paying and that until that time it has paid 12 coins (0CH) of type 1, still having to pay 24 coins (18H) and which has paid 33 coins (21H) of type 2, still having to pay 11 (0BH)

1.6.3.3. Cancellation of a command in execution [15H = 21D]



This command causes the cancellation of the command that is being executed at that time. If the Hopper is on standby, it responds with a negative acknowledgement string.

If the Hopper is on 'Standby':

Send: **[Dir] [00] [01] [15H] [Chk]**

Reply: **[01] [00] [Dir] [05] [Chk]**

Where:

Dir = address of the corresponding Hopper

If the Hopper is 'Emptying':

Send: **[Dir] [00] [01] [15H] [Chk]**

Reply: **[01] [03] [Dir] [00] [Data 1] [Data 2] [Data 3] [Data 4] [Data 5] [Chk]**

Where:

Data 1 = 24H = Hopper emptying

Data 2/3 = High/low part of the counter of coins type 1 paid

Data 4/5 = High/low part of the counter of coins type 2 paid

Dir = address of the corresponding Hopper

If the Hopper is 'Paying':

Send: **[Dir] [00] [01] [15H] [Chk]**

Reply: **[01] [05] [Dir] [13H] [Data 1] [Data 2] [Data 3] ... [Data 9] [Chk]**

Where:

Data 1 = 25H = Hopper paying

Data 2/3 = High/low part of the number of coins type 1 paid

Data 4/5 = High/low part of the number of coins type 1 to be paid

Data 6/7 = High/low part of the number of coins type 2 paid

Data 8/9 = High/Low part of the number of coins type 2 to be paid

Dir = address of the corresponding Hopper

Example:

The Machine requests the Hopper number 3 to cancel the command that it is executing at that time.

Send: [05] [00] [01] [15] [E5H]

If the Hopper is on 'Standby':

Reply: [01] [00] [05] [05] [F5H]

If the Hopper is 'Emptying':



Reply: [01] [05] [05] [13H] [24H] [00] [23H] [00] [12H] [89H]

The Hopper has cancelled an emptying command, at the time of the cancellation 35 coins (0023H) type 1 had been extracted, and 18 coins (0012H) type 2.

If the Hopper is 'Paying':

Reply: [01] [09] [05] [13H] [25H] [00] [01] [00] [02] [00] [02] [00] [06]
[AEH]

The Hopper has cancelled a payment command and at the time of the cancellation, 1 coin of type 1 had been paid, still having to pay 2 coins, and 2 coins of type 2 had been paid, still having to pay 6 coins.

1.6.3.4. Emptying [19H = 25D]

This command causes the complete emptying of the Hopper. Before being able to use this command, the Hopper must have been enabled by means of the **[Enable Hopper]** command [A4 H = 164 d].

In the event that the Hopper can execute the command, it returns a positive acknowledgement string and begins to extract coins until a maximum span is detected, or the cancellation command is received or an error is detected or a reset hardware (power supply fault) is carried out.

In the event that the Hopper cannot execute the command, it returns a negative acknowledgement string and continues in the status it was. This negative acknowledgement being sent may be due to the Hopper being disabled, or that it is on error, or that another command is being executed at that time.

Send: [Dir] [00] [01] [19H] [Chk]

Reply: [01] [00] [Dir] [00] [Chk] -> Positive acknowledgement

Reply: [01] [00] [Dir] [05] [Chk] -> Negative acknowledgement

Where:

Dir = address of the corresponding Hopper

Example:



The Machine requests hopper 2 to empty all the coins.

Send: [04] [00] [01] [19H] [20H] [Data 1] [Data 2] ... [Data 7] [Chk]

Reply: [01] [00] [04] [00] [19] [xx] [yy] [zz] [mm] [Chk]

This reply indicates that the Hopper has begun to execute the emptying command

1.6.3.5. Payment [20H = 32D]

This command causes the payment of the number of coins type 1 and type 2 indicated in bytes 'Data 4' and 'Data 5' respectively. The bytes Data 1, Data 2 and Data 3 correspond to the Serial number of the hopper (The byte Data 1 is the LSB and Data 3 is the MSB).

The reading of the Serial number is done using the **[Request Serial Number]** command [F2H = 242d].

Before making a payment, the hopper should be enabled using the command **[Enable Hopper]** [A4 H = 164 d].

In the event that the Hopper can execute the command, it returns a positive acknowledgement string and begins to make the payment until the coins indicated have been extracted, or a maximum span is detected, or the cancellation command is received or an error is detected or reset hardware (power fault) is detected.

If the Hopper cannot execute the command, it returns a negative acknowledgement string and continues in the status it was.

This negative acknowledgement being sent may be due to the Hopper being disabled or the safety code is not correct, or that it is on error, or that it is executing another command at that time.

Send: [Dir] [07] [01] [20H] [Data 1] [Data 2] ... [Data 7] [Chk]

Reply: [01] [00] [Dir] [00] [Chk] -> Positive acknowledgement

Reply: [01] [00] [Dir] [05] [Chk] -> negative acknowledgement

Where:

Dir = address of the corresponding Hopper

Example:

The Machine requests the Hopper to make a payment of 8 coins of type 1 and 5 coins of type 2.



Send: [03] [07] [01H] [20H] [4EH] [61H] [BCH] [00H] [08H] [00H] [05H] [5DH]

Reply: [01] [00] [03] [00] [FCH]

This reply indicates that the Hopper has started to execute the payment command

1.6.3.6. Last command status [20H = 32D]

Command that requests information from the Hopper about the 'Last command' executed and about the parameters of this command.

Three possibilities of 'Last command' are considered. These are Standby, Emptying and Paying.

If the 'Last command' executed was Emptying, the number of coins extracted is also included (2 bytes).

Send: [Dir] [00] [01] [23H] [Chk]

Reply: [01] [05] [Dir] [00] [Data 1] [Data 2] [...] [Data 5] [Chk]

Where:

Data 1 = 24H = last command executed was the Emptying command

Data 2/3 = High/low part of the counter of coin type 1 extracted

Data 4/5 = High/low part of the counter of coins type 2 extracted

Dir = Address of the corresponding Hopper.

If the last command executed was the Payment command, the number of coins paid (2 bytes) and the number of coins to be paid (2 bytes) are also included.

Send: [Dir] [00] [01] [23H] [Chk]

Reply: [01] [09] [Dir] [00] [Data 1] [Data 2] [...] [Data 9] [Chk]

Where:

Data 1 = 25H = Last command executed was the Payment command

Data 2/3 = High/low part of the counter of coins type 1 paid

Data 4/5 = High/low part of the number of coins type 1 to be paid

Data 6/7 = High/low part of the counter of coins type 2 paid

Data 8/9 = High/Low part of the number of coins type 2 to be paid

Dir = address of the corresponding Hopper

This request for status informs of the status at the end of the last command executed by the Hopper, but not how this command ended, that is, if the last command executed was a



Payment command of 10 coins of type 1 and 3 of type 2, we know that the command finished having paid 6 of the 10 coins of type 1 and 1 of the 3 coins of type 2, but it does not give information about if the termination was due to a maximum span being reached or if a cancellation command of the command was sent.

Example:

The machine requests the Hopper number 1 information about the last command executed.

Send: [03] [00] [01] [23H] [D9H]

If the Hopper was on 'Standby':

Reply: [01] [01] [03] [00] [01] [FAH]

This reply indicates that the Hopper has not executed any command since it was put into service

If the Hopper executed an Emptying command:

Reply: [01] [05] [03] [00] [19H] [00] [04H] [01] [54H] [85H]

This reply indicates that the last command executed was the emptying command and that 4 coins (0004H9) of type 1 and 340 coins (0154H) of type 2 were extracted

If the Hopper executed a Payment command:

Reply: [01] [09] [03] [00] [20H] [00] [02] [00] [07] [00] [03] [00] [08] [BFH]

This reply indicates that the last command executed was the payment command and that, of type 1, 2 coins were paid, 7 still having to be paid, and of type 2, 3 coins were paid, still having to pay 8

1.6.3.7. Diameter programming [25H = 37D]

With this command, the diameters of the coins that the discriminator is going to work with are programmed. This command has 4 data bytes, with which the average diameter (in tenths of



a millimetre) is indicated of coins type 1 and type 2, respectively.

If the Hopper can execute the command, it returns a positive acknowledgement string and updates the parameters of the coins.

If the Hopper cannot execute the command, it returns a negative acknowledgement string and continues in the status that it was in.

Send: **[Dir] [04] [01] [25H] [Data 1] [Data 2] [...] [Data 4] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk]** -> Positive acknowledgement

Reply: **[01] [00] [Dir] [05] [Chk]** -> Negative acknowledgement

Where:

Dir = address of the corresponding Hopper

Data 1/2: High/low part of the average diameter of the coin type 1

Data 3/4: High/low part of the average diameter of the coin type 2

Example:

The Machine programs the coin 1 with an average diameter of 20.4 millimetres and coin 2 with an average diameter of 26.0 millimetres.

Send: **[03] [04] [01] [25] [00] [CC] [01] [04] [07]**

Reply: **[01] [00] [03] [00] [FC]**

This reply indicates that the Hopper has executed the command

1.6.3.8. Request for parameters [29H = 41D]

With this command, the Discriminator is requested to notify the values of the diameters of the coins that are programmed. This reply is comprised of 4 data bytes, with which the average diameter (in tenths of a millimetre) of the coins type 1 and type 2, are indicated, respectively.

If the Hopper can execute the command, it returns a string with the values requested.

If the Hopper cannot execute the command, it returns a negative acknowledgement string and continues in the status it was.

Send: **[Dir] [00] [01] [29H] [Chk]**

Reply: **[01] [04] [Dir] [00] [Data 1] [Data 2] [...] [Data 4] [Chk]** -> Positive acknowledgement

Reply: **[01] [00] [Dir] [05] [Chk]** -> Negative acknowledgement

Where:



Dir = address of the corresponding Hopper

Data 1/2: High /low part of the average diameter of the coin type 1

Data 3/4: High/low part of the average diameter of coin type 2

Example:

The Machine requests the Hopper to return the parameters that were programmed.

Send: [03] [00] [01] [29] [D3]

Reply: [01] [08] [03] [00] [00] [EE] [01] [1F] [E6]

The average diameter of the standard coin is 23.8 millimetres and that of coin 2 is 28.7 millimetres

1.6.4. INTELLIGENT PAYOUT

Intelligent payout is a new concept that is exploited by the DISCRIMINATOR hopper range. The Machine communicates to the hopper the amount of money it must return and the hopper decides which coins of those it has programmed it will use to make the payment. Therefore, it is necessary to programme the value of each one of the coins the hopper is going to work with.

Working under "intelligent payout" the hopper will always pay what it can with the coins of higher value so as not to be left without the coins of lower value and therefore "out of change".

The values of the coins will be given in relation to the base coin. So the corresponding values of the coins €2, €1, 50c, 20c, 10c y 5c, if the base coin is 1c, will be 200, 100, 50, 20, 10 y 5 respectively.

Before programming the value of the coins, it is necessary to programme their values.

The information of all the programmed values is stored in the eeprom.

1.6.4.1. Programming the value [31H = 49d]

With this command, we programme the coin values.

Send: **[Dir] [04] [01] [31] [Data 1] ... [Data 4] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk] ->ACK**
[01] [00] [Dir] [05] [Chk] ->NACK

Where: Dir = Hopper address
Data 1 / 2 = high / low part of the value of coin type 1
Data 3 / 4 = high / low part of the value of coin type 2



1.6.4.2. Request value [32 H = 50d]

With this command, we request the hopper for the values it has programmed.

Send: **[Dir] [00] [01] [32] [Chk]**
Reply: **[01] [04] [Dir] [00] [Data 1] ... [Data 4] [Chk]**
Where: Dir = Hopper address
 Data 1 / 2 = high / low part of the value of coin type 1
 Data 3 / 4 = high / low part of the value of coin type 2

1.6.4.3. Intelligent payout [35 H = 53d]

With this command, we order the payment of a number of base coins.

Send: **[Dir] [05] [01] [35] [Data 1] ... [Data 5] [Chk]**
Reply: **[01] [00] [Dir] [00] [Chk] ->ACK**
 [01] [00] [Dir] [05] [Chk] ->NACK
Where: Dir = hopper address
 Data 1 / 2 / 3 = serial number of the hopper (Data 1 = LSB)
 Data 4 / 5 = low / high part of the number of base coin to pay out

1.6.4.4. Status of an intelligent payout while paying [13H = 19d]

With this command, we request the hopper to inform on its status of the intelligent payout command presently in operation.

Send: **[Dir] [00] [01] [13] [Chk]**
Reply: **[01] [09] [Dir] [35] [Data 2] ... [Data 9] [Chk]**
Where: Dir = hopper address
 Data 2 / 3 = high / low part of the number of base coins paid out in the last payment.
 Data 4 / 5 = high / low part of the number of base coins still to pay out in the last payment.
 Data 6 / 7 = high / low part of the number of coins type 1 paid out.
 Data 8 / 9 = high / low part of the number of coins type 2 paid out.

1.6.4.5. Status of an intelligent payout after paying [23H = 35d]

With this command, we request the hopper to inform on its status of the intelligent payout command just finished.



Send: **[Dir] [00] [01] [23] [00] [Chk]]**

Reply: **[01] [09] [Dir] [35] [Data 2] ... [Data 9] [Chk]**

Where: Dir = hopper address

Data 2 / 3 = high / low part of the number of base coins paid out in the last payment.

Data 4 / 5 = high / low part of the number of base coins still to pay out in the last payment.

Data 6 / 7 = high / low part of the number of coins type 1 paid out.

Data 8 / 9 = high / low part of the number of coins type 2 paid out.

1.6.5. MDCES (Multi-Drop Command Extension Set)

By default, the Hopper is assigned the addresses that go from 3 to 6, being able to assign any of these to a particular Hopper by means of its corresponding dipswitch.

However, these addresses (3-6) may be used in turn by any other type of device (Selector, Hopper...) so conflicts may be caused. cctalk® permits the machine to modify the address of any slave device by means of 4 commands that are explained below.

However, some of the MDCES commands do not adapt to the standard structure of cctalk®.

Once the Machine has modified the default address of a Hopper, this keeps the new address and ignores the address indicated by its dipswitch. If the Hopper uses the address indicated by the dipswitch again, the Machine must inform of this previously by programming its address as address 0.

1.6.5.1. Address poll [FDH = 252D]

By means of this command, all the slave devices are requested to return their addresses. To do this, it is sent with destination address 0 (Broadcast). In order to avoid collisions, the address byte is returned with a delay proportional to the address value.

Send: **[00] [00] [01] [FDH] [02]**

Reply: **{Variable delay} [Dir]**

Where:

Dir = address of the corresponding Hopper.

The algorithm to calculate the delay with which it has to reply is as follows:

Disable port rx

Delay (4*Dir) ms

Send [Dir]



Delay 1200 (4*dir) ms
Enable port rx

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

1.6.5.2. Address clash [FCH = 252D]

This command is used to check if one or more Hoppers share the same address. Unlike the Address Poll command, this is sent to a specific address.

In order to avoid collisions, the address byte is returned with a delay proportional to the address value:

Send: **[Dir] [00] [01] [FCH] [Chk]**
Reply: **{Variable delay} [Dir]**

Where:

Dir = address of the corresponding Hopper

The algorithm to calculate the delay with which it has to reply is as follows:

R=rand (256)
Disenable port rx
Delay (4*Dir) ms
 Send [Dir]
 Delay 1200 (4*dir) ms
 Enable port rx

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

Although the possibility exists of two devices that share the same address generating the same random number, the likelihood of this occurring is very small (1 out of $254 \times 256 = 1$ out of 65,024).

1.6.5.3. Address change [FBH = 251D]

This command permits reprogramming the Hopper with a new address, valid for all the commands it receives from now on.

If the new address is number 0, the Hopper will obtain its new address (only addresses 3 to 6) from its corresponding dipswitch; however, the Machine will not know which this new address is.

Therefore, once this command has been used with this option, the Machine should send the



[Address Poll] command.

Send: **[Dir] [01] [01] [FBH] [Data 1] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk]** -> Positive acknowledgement

Where:

Dir = address of the corresponding Hopper

Data 1 = new address of the corresponding Hopper

Example:

The machine programs the Hopper whose address is 3 with address 13.

Send: **[03] [01] [01] [FB] [0D] [F3]**

Reply: **[01] [00] [03] [00] [FC]**

The machine programs the Hopper whose address is 13 in order to obtain its new address from its dipswitch.

Send: **[0D] [01] [01] [FB] [00] [F6]**

Reply: **[0D] [00] [03] [00] [F0]**

The Hopper whose address is 13 has changed its address and now it has the one indicated by its dip switch, but, the machine, in principle, does not know what this address is

1.6.5.4. Address random [FAH = 250D]

This command permits reprogramming the Hopper with a new address whose value will be random. This is an escape channel for cases when several devices share the same address asking afterwards for its new addresses. After this command, the machine must send the Address Poll command to know the values of the new addresses.

Send: **[Dir] [01] [00] [FAH] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk]** -> Positive acknowledgement

Where:

Dir = address of the corresponding Hopper

Example:

The machine notifies the Hoppers whose addresses are number 3 that they should



reprogram them with random values.

Send: [03] [01] [00] [FA] [02]

Reply: [01] [00] [03] [00] [FC]

1.6.6. Calibration commands

1.6.6.1. Calibration [43H = 67D]

This command is similar to an **[Empty]** [19H = 25 d] with the difference that in this case no anti span is carried out so that the process can be carried out faster. A simple way to make the hopper not extract coins during the calibration process is to first send the **[parameter programming]** command [25H = 37 d] with zero diameters.

Before sending the **[Calibration]** command the hopper must have first been activated with the **[enable hopper]** command [A4 H = 164 d].

If the Hopper can execute the command, it returns a positive acknowledgement string and starts to measure the coins until it detects a cancellation command or an error is detected or a hardware reset is carried out.

If the Hopper cannot execute the command, it returns a negative acknowledgement string and continues in the state it was. Sending this negative acknowledgement may be due to the Hopper being disabled, or in error, or that another command is being executed at that time.

Send: **[Dir] [00] [01] [43H] [Chk]**

Reply: **[01] [00] [Dir] [00] [Chk]** -> Positive acknowledgement

Reply: **[01] [00] [Dir] [05] [Chk]** -> negative acknowledgement

Where:

Dir = address of the corresponding Hopper

1.6.6.2. Request for impulses [41H = 65D]

With this command, the Discriminator is requested to notify the range of values of the diameters of the coins that it has programmed. This reply is comprised of 8 data bytes, with which it indicates the **minimum and maximum diameter** (in number of impulses of the motor encoder) of coins types 1 and 2, respectively.

Send: **[Dir] [00] [01] [65H] [Chk]**

Reply: **[01] [08] [Dir] [00] [Data 1] [Data 2] [...] [Data 8] [Chk]** -> positive acknowledgement

Reply: **[01] [00] [Dir] [05] [Chk]** -> negative acknowledgement

Where:



Dir = address of the corresponding Hopper

Data 1 / 2: High/low part of the minimum diameter of coin type 1

Data 3 / 4: High/low part of the maximum diameter of coin type 1

Data 5 / 6: High/low part of the minimum diameter of coin type 2

Data 7 / 8: High/low part of the maximum diameter of coin type 2

1.6.6.3. Request measurement [40H = 64D]

With this command, the Discriminator is requested to notify the values of the diameters of the last coin that it has measured. Apart from the last coin, it also sends the cavity where this measurement has been made. Once the information has been transmitted, the hopper resets the value of the last measurement made so that if the **[request measurement]** command is resent the value will be 0 until the hopper measures again.

Send: **[Dir] [00] [01] [64H] [Chk]**

Reply: **[01] [03] [Dir] [00] [Data 1] [Data 2] [Data 3] [Chk]** Reply: **[01] [00]**
 [Dir] [05] [Chk]

Where:

Dir = address of the corresponding Hopper

Data 1 / 2: High/low part of the diameter of the coin in pulses of the encoder

Data 3: Cavity in which the measurement was made

1.6.6.4. Offset programming [42H = 66D]

With this command, the suitable calibration is programmed in the discriminator. On the one hand, the diameter of the coin used for the calibration (in tenths of a mm) is programmed and also the average value in encoder pulses of a large number of the same coin type. With this data the manufacturing differences between the different hoppers is compensated. On the other hand, the differences between the cavities on a disk are compensated by programming the cavity offsets. To do this a number of readings are taken with a number of coins of a particular type in all the cavities on the disk. These average values, one for each cavity, are all different so you look for the lowest and the difference in relation to all the others is found. These differences together with the corresponding cavity are the values to programme. The offset of the cavity with the lowest value is also programmed. This offset will be 0 so the hopper will use this cavity as a reference to compensate the values in the rest.

The programming of the total offset and the programming of the cavity offsets is done in the same command. When programming the total offset the first two bytes correspond to the diameter in tenths of a mm of the coin used while in programming cavity offsets these two



values correspond to the cavity you wish to programme. The minimum value of the diameter of a coin is 180 tenths of a mm and the maximum value of a cavity is 8 so these two first bytes also indicate if you wish to programme the total or cavity offsets. The next two bytes correspond to the average value of a set of coin measurements or the cavity offset.

Send: **[Dir] [04] [01] [66H] [Data 1] [Data 2] [Data 3] [Data 4] [Chk]**
Reply: **[01] [03] [Dir] [00] [Chk]**

Where:

Dir = address of the corresponding Hopper

Data 1 /2: High/low part of the coin diameter or the cavity number

Data 3 /4: High/low part of the offset to be programmed (total or cavity)

1.6.6.5. Algorithm and sequence of commands for calibration

- Enable the Hopper by sending **[Enable hopper]** [A4H] command.
- Collect and save the diameters programmed in the Hopper with the **[Diameter request]** [29H].
- Set the diameters to 0 by sending the **[Diameter Programming]** [25H] command.
- Set the offsets of all the cavities and the total offset to 0 by sending an **[Offset Programming]** [42H] command sequence.
- Send the **Calibration** [43H] command and the Hopper will start up. It is advisable to fill the hopper half-full (about 20 coins of €1) to do the calibration.
- Now a loop starts by continuously sending the **Request for State** [13H] and **Request for measurement** [40H] commands. The first will give us information about whether the Hopper continues calibrating and the second informs us of the last coin measured and the cavity it was. For each cavity all the measurements other than 0 must be kept in a table, until the number of measurements that you wish to carry out per cavity has been completed (it is advisable to collect at least 25 measurements or more per cavity).

The loop will end when all the measurements have been obtained for all the cavities, and the Hopper will send the **[Cancellation]** [15H] command.

- Restore the diameters that were programmed previously with the **Parameter programming** [25H] command.
- Then the calibration calculations will be made with the measurements collected. The measurement tables of each cavity must be put in order and 20% of the smaller measurements and 20% of the larger ones must be rejected. In this way we reject the



deviations over the average. With the resulting data, calculate the mean of each cavity.

- The offsets to be sent to the Hopper will be calculated as follows: Look for the minimum of the averages of the 8 cavities; this will be the total offset. The offset for each cavity will be calculated by deducting the total offset from the mean of each cavity. All these offsets will be sent with the **Offset Programming** [42H] command.
- After reaching this point, the Hopper will be correctly calibrated and you will be able to continue with the normal working of the machine.

Table 2: POSSIBLE HOPPER STATES

The possible hopper states are indicated in the table.

This information is obtained by a Status request by the machine.

Code	State	Data Bytes in strings Rode U – Machine
01H	Standby	1 byte
02H	Error	2 bytes: Error code
24H	Empty	5 bytes
25H	Payment	9 bytes

The first of the data bytes sent, Data 1, corresponds to the status of the hopper.

- (1) Data 2 and 3 the high / low part of the number of coins of type 1 extracted.
Data 4 and 5 the high / low part of the number of coins of type 2 extracted.
- (2) Data 2 y 3 the high / low part of the number of coins of type 1 paid.
Data 4 y 5 the high / low part of the number of coins of type 1 still to pay.
Data 6 y 7 the high / low part of the number of coins of type 2 paid.
Data 8 y 9 the high / low part of the number of coins of type 2 still to pay.

Table 3: POSSIBLE LAST COMMANDS OF THE HOPPER

The table shows the possible last commands a hopper could have executed.

This information is obtained with the Status last command request by the machine.

Code	Status	Data Bytes in strings Hopper – Machine
01H	Standby	1 byte
24H	Emptying	5 bytes



25H	Paying	9 bytes
-----	--------	---------

The first of the data bytes sent, Data 1, corresponds to the status of the hopper.

- (1) Data 2 and 3 the high / low part of the number of coins of type 1 extracted,
Data 4 and 5 the high / low part of the number of coins of type 2 extracted.
- (2) Data 2 and 3 the high / low part of the number of coins of type 1 paid out,
Data 4 and 5 the high / low part of the number of coins of type 2 still to pay,
Data 6 and 7 the high / low part of the number of coins of type 1 paid out,
Data 8 and 9 the high / low part of the number of coins of type 2 still to pay.

Table 4: ERRORS

Code	Description
01H	Coin exit detector permanently active
02H	Coin exit detector during standby
04H	Jam in motor permanent
08H	Motor does not respond
10H	Fault in hardware coin exit detector
20H	Fault in photodiode coin exit detector
40H	Fault in encoder opto
80H	Fault in trigger opto



Brands of



AZKOYEN

AZKOYEN MEDIOS DE PAGO S.A.



Teidde