



HOPPER - U



1. SERIAL cctalk COMMUNICATION PROTOCOL

The communication protocol fitted into the Hopper "U" 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. It has been designed to permit the interconnection of different coin mechanisms (Change dispensers, coin validators, etc.) in one single bus with two wires (one bi-directional data line and the mass).

The use of cctalk® is open, and it therefore can be used without paying for licenses or fees.

1.1. COMMUNICATION TOPOLOGY

The data are established with the levels TTL : '0' logic 5V, '1' logic 0v and Status the standby '1' logic.

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, 1 stop bit

The message format is as follows:



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

Each communication sequence is comprised of two rasters. The first corresponds to the command sent by the machine to the Change dispenser and the second is the reply sent by the Change dispenser to the machine. Both parts have the format indicated above.

1.2.1. Destination address

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

- 0:** Used in messages that affect all the Change dispensers at the same time.
- 1:** Address of the machine.
- 2:** Address of the Change dispenser in communication with one single Change dispenser.
- 3 to 255:** Address of the Change dispensers in multi-Change dispenser communications.

The 'Destination address' byte indicates the bus node the message is going to be addressed to.

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

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

1.2.2. Number of data bytes

The range of the number 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.

1.2.3. Source address

The range of addresses goes from 1 to 255 (of which 254 correspond to Change Dispenser addresses).

- 1:** Machine address.
- 2:** Address of the Change Dispenser in communications with one single Change Dispenser.
- 3 to 255:** Addresses of the Change Dispensers in multi-Change dispenser 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 Change dispenser configured with the dip: switch such as 1, 2, 3 or 4, respectively.
- 3 to 255:** When the message is sent by the Change dispenser configured by means of software with the **MDCES** commands, and with its address saved in the EEPROM.

1.2.4. 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 Change dispensers, the Header will have '0' value in all the messages, except in the 'Negative Acknowledgement' or NACK message.

1.2.5. 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.

1.2.6. Checksum

Checksum is what makes the 8 less important bits of the sum of all the bytes of the message,



including the actual checksum, give '0' as the result.

Example:

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

$$1 + 0 + 2 + 0 + 253 = 256 = 0$$

1.3. TIME REQUIREMENTS

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

1.3.1. Maximum time between command and reply

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

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

1.3.2. Minimum time from the supply until the first command is sent

The minimum time that must elapse from the time the Change dispenser is supplied until the first command is sent must be 200 milliseconds.

1.4. ERROR MANAGEMENT

1.4.1. Actions faced with an error

If a Change dispenser receives an incomplete message (receipt timeout) or with an incorrect checksum, the only action is carries out is to initialise 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.

The advisable time for waiting to send another attempt will be between 75 and 100 milliseconds

On the other hand, if the machine receives an incomplete message (receipt timeout) or with an incorrect checksum, it can choose to resend the same message. In any case, faced with an error in receipt of a message no negative acknowledgement message has been defined, which simplifies the implementation of multi-Change dispenser protocols and reduces collisions.

If a Change dispenser 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 or emptying command is received.

Communication of errors

The Change dispenser carries out a series of controls of 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 error(s) by means of the "Request for Status" command requested by the latter from the Change dispenser. On this Request the Change Dispenser 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 relative error has been detected and the ones 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 Change dispenser 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 Change Dispenser will enter the Error status again.

1.5. COMMANDS

Within the commands implemented, there are some that are specific of cctalk® and there are others that are particular of the hopper Change Dispenser. Out of the specific commands of cctalk®, there is a first group which is general, that is, valid commands for any type of device. The other specific commands of cctalk ® are designed to be used with Change Dispensers. In Table the codes of the commands implemented are indicated.



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
C0H	Request built code	0	N
F6H	Request manufacturer ID	0	7
F1H	Request Software Revision	0	2
ECH	Read opto states	0	1
04H	Request Comms Revision	0	3
01H	Reset Device	0	1
F2H	Request serial number	0	3
DBH	Enter new PIN number	4	0
DAH	Enter pin number	4	0
Typical commands of cctalk®. Commands for Change Dispenser			
A4H	Enable hopper	1	0
A7H	Dispense hopper coins	4	0
A6H	Request hopper status	0	4
A3H	Test hopper	0	1
ACH	Emergency stop	0	1
A8H	Request hopper dispense count	0	3
Typical commands of cctalk®. Multi-Drop Command Extension Set (MDCES)			
FDH	Address Poll	0	Sólo [Dir]
FCH	Address Clash	0	Sólo [Dir]
FBH	Address Change	1	0
FAH	Address Random	0	0



1.5.1. Specific commands of cctalk®. General commands

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

1.5.1.1. Simple pollLL [FE H = 254 D]

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 (Receipt timeout in machine) it will indicate that the relative 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 encoded as number 2 in the bus.

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

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

Send: **[05] [00] [01] [FEH] [FCH]**
Reply: **No answer**

1.5.1.2. Request equipment category ID [F5 H = 245 D]

This command permits receipt from the relative hopper of the chain of characters that identifies the type of device in question.

In the case of hopper, which is the device in question, 'Payout' 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 relative hopper

Example:

The machine requests dispenser 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] [50H] [61H] [79H] [6FH] [75H] [74H] [74H]

Bytes [50H] [61H] [79H] [6FH] [75H] [74H] correspond to characters 'P','a','y','o','u' and 't' respectively

1.5.1.3. Request product code [F4 H = 244 D]

With this command, the relative 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, the chain of characters to be transmitted still has not been decided.

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 = Number of data to be sent
Dir = address of the relative hopper.

Example:

The machine requests the Hopper number 3 its product code, which in this example is "Payout"

Send: [05] [00] [01] [F4H] [06]
Reply: [01] [06] [03] [00] [50H] [61H] [79H] [6FH] [75H] [74H] [74H]

The bytes [50H] [61H] [79H] [6FH] [75H] [74H] [74H] correspond to characters 'P','a','y','o','u' and 't' respectively

1.5.1.4. Request build code C0 H = 192 D]

With this command, the relative hopper sends the machine the device assembly code. 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, the chain of characters to be transmitted still has not been decided.

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 = Number of data to be sent Dir =



address of the relative hopper.

Example:

The machine requests the Hopper number 3 its product code, which in this example is "Payout"

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

Reply: [01] [06] [03] [00] [50H] [61H] [79H] [6FH] [75H] [74H] [74H]

The bytes [50H] [61H] [79H] [6FH] [75H] [74H] [74H] correspond to characters 'P', 'a', 'y', 'o', 'u' and 't' respectively

1.5.1.5. Request manufacturer ID [F6 H = 246 D]

With this command, the relative hopper sends the machine the identification of the device manufacturer. In the case of the hopper 'U', 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 relative 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

1.5.1.6. Request software revision [F1 H = 241 D]

With this command, the relative hopper sends the machine the current software version of the device.

cctalk® does not establish any restriction concerning the format of the reply message.

Send: [Dir] [00] [01] [F1H] [Chk]

Reply: [01] [02] [Dir] [00] [Data 1] [Data 2] [Chk]

Where: Data 1 = Byte prior to the dot of the program version Data 2 = Byte following



the dot of the program version Dir = address of the relative hopper.

Example:

The machine requests the hopper number 3 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 informs the machine that the software version is 9.2

1.5.1.7. 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.

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 relative 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
½ Load	0	1
Full	0	0

Example:

The machine requests the hopper number 2 the value of the empty, full and scale detectors.

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

From this reply it is deduced that the Empty detector=1 ,Full detector=0 and scale detector=1



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

As a reply to this command, the **hopper** sends the implementation level of the **cctalk®** protocol (01H in our case) and the communication software version. If this version is 3.6, the 'Greater revision' will be 3 and the 'Lesser 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 relative "hopper".

Example:

The machine requests the hopper number 1 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] [03] [01] [F4H]**

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 3.1.

1.5.1.9. Reset device [01 H = 1 D]

This command causes the hopper to carry out a software reset. At that time the execution of the program of the hopper goes to the reset vector.

The affected hopper sends a positive acknowledgement raster immediately before making the reset. After a reset there will be a minimum waiting time of 250 milliseconds before sending the next command.

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]**

1.5.1.10. 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 the hopper number 1 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

1.5.1.11. Enter new PIN number [DB H= 219 D]

All commands implemented on the payout unit can be programmed by PIN number. The current PIN number can be changed using this command, but since this command is also protected by a PIN number, the current PIN number must be entered first.

If a cctalk command is executed without having sent the correct PIN number (included in this command), there will be no response by the Hopper.

The PIN number is binary code of 32 bits (4,294,967,296 combinations).

By changing the PIN number to a value of 0, protection by the PIN is disabled.

Send: [Dir] [00] [01] [DB H] [PIN 1] [PIN 2] [PIN 3] [PIN 4] [Chk]
Reply: [01] [03] [Dir] [00] [Chk]

1.5.1.12. Enter PIN number [DA H= 218 D]

All commands implemented on the payout unit can be protected by a PIN number. The current PIN number can be entered using this command. If the current PIN number is other than 0, it must be entered every time after the electric power is connected and after each reset in order to be able to execute a command.

If a cctalk command is executed without having sent the correct PIN number, there will be no response by the Hopper.

Whether the PIN number entered is correct or incorrect, the payout unit will send an affirmative response.

Send: [Dir] [00] [01] [DB H] [PIN 1] [PIN 2] [PIN 3] [PIN 4] [Chk]
Reply: [01] [03] [Dir] [00] [Chk]



1.5.2. Comandos específicos de cctalk. Comandos para devolvedores

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

1.5.2.1. Enable hopper [A4 H = 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 relative hopper Data1 = [A5H] hopper enabled Other than [A5H] hopper disenabled.

Example:

The machine enables the hopper 1.

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

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

1.5.2.2. Dispense hopper coins [A7 H = 167 D]

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 raster and begins to make the payment until the coins indicated have been extracted, or a maximum spacing 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 raster and continues in the status it was in. 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 relative 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 the 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

1.5.2.3. Request hopper status [A6 H = 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 hopper 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] [Dato 1] [Dato 2] [Dato 3] [Dato 4] [Chk]

Where: Dir = address of the relative 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 paid during last payment

The information of this command is kept at all times in the EEPROM, so, in spite of there being a supply fault, we will be able to recuperate the status of the last command when the Hopper is reset.

On sending a Reset Device the last status is updated and it goes on to Standby.

Example:

The machine requests the 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

1.5.2.4. Test hopper [A3 H = 163 D]

This command causes information to be sent about several error and operation flags of the Hopper. The flags keep information about the event occurred until enquiries are made about them, and after passing the information to the machine they are reset.

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

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

Where: Dir = address of the relative 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 blockage.

 Bit 3 Opto fraud attempt, release blocked during standby.

 Bit 4 Opto fraud attempt, short-circuit during standby. (NOT IMPLEMENTED)

 Bit 5 Opto blocked during payment.

 Bit 6 Hardware reset occurred (NOT IMPLEMENTED)

 Bit 7 Payment disenabled.

 1 = True, 0 = False.

Example:

The machine requests the hopper 1 information about several parameters.

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

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

The hopper has undergone an excess maximum absolute current (probably due to a blockage), during the last payment the motor had to be turned in the opposite direction to undo a blockage and the payment has been disenabled

1.5.2.5. Emergency stop [AC H = 172 D]

This command automatically stops the payment sequence, due to the Dispense Hopper Coins or Payment commands, and transmits the number of coins of type 1 that still have to be paid.

In the event that a payment is being made, the reply will be the same, but with coins that still have to be paid on 0. This command is also used to make the emergency stop of an Emptying,



a specific command of this Hopper, and it is also answered with an 0 in coins that still have to be paid.

Unlike the rest of the commands the reply to this will be received by the machine 300 milliseconds afterwards due to its processing time.

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

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

Where: Dir = address of the relative 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]**

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

1.5.2.6. Request hopper dispense count [A8 H = 168d]

This command causes the hopper to 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 relative hopper

 [Data 1] = Number of coins of type 1 paid (LSB)

 [Data 2] = Number of coins of type 1 paid.

 [Data 3] = Number of coins of type 1 paid (MSB).

Example:

The machine requests the hopper 1 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 by the hopper 1 has been 12,810 since the last reset



1.5.3. 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 relative dip switch.

However, these addresses (3-6) may be used in turn by any other type of device (Selector, Change Dispenser...) 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 dip switch. If the hopper uses the address indicated by the dip switch again, the machine must inform of this previously by programming its address as address 0.

1.5.3.1. Adress poll [FDH]

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, only the address byte with a proportional delay to the value of the address is returned.

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

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

Where: Dir = address of the relative hopper.

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

Disenable port rx
Delay (4*Dir) milliseconds
Send [Dir]
Delay 1200 (4*dir)milliseconds
Enable port rx

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

1.5.3.2. Adress clash [FCH]

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 only returned with a random delay in the



sending.

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

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

Where: Dir = address of the relative 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) miliseconds  
Send [Dir]  
Delay 1200 (4*dir)miliseconds  
Enable port rx
```

If the machine receives all the bytes about 1.5 seconds 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).

1.5.3.3. Adress change [FBH]

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 relative dip switch, however, the machine will not know which is this new address. Therefore, once this command has been used with this option, the machine should send the Address Poll command.

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

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

Where: Dir = address of the relative hopper

 Data1 = new address of the relative 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 dip switch.

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

1.5.3.4. Adress random [FAH]

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 relative hopper

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

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]



Brands of



AZKOYEN

AZKOYEN MEDIOS DE PAGO S.A.



Teidde