AZKOYEN
MEDIOS DE PAGO

# CCTALK PROTOCOL SPECIFICATION

# T3 HOPPER

AZKOYEN Medios de Pago, S.A
Avda. San Silvestre, s/n
31350 Peralta (Navarra) Spain

Tel.: +34 948 709 709
Fax: +34 948 709 709
www.azkoyen.com

# INDEX OF CONTENTS

# 1 GENERAL DESCRIPTION OF THE CCTALK PROTOCOL

The communication protocol implemented in the Hopper T3 range is compatible with ccTalk®.

The specification used for the implementation of this Protocol is the following:

**ccTalk Serial Communication Protocol**
**Generic Specification**

**Issue 4.5**

Cctalk® is a serial communication protocol, developed by Coin Controls Ltd. for low transmission velocity control networks. It has been designed to allow the interconnection of different coin mechanisms (Hopper, validators, etc.) on a simple bus of two lines (one line for bidirectional data and the other ground).

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**

## 1.1 COMMUNICATION TOPOLOGY

The communication topology is similar to the RS232 standard but without negative voltages. The communications are stablished through a bi-directional, single line, for transmission and reception tasks.

The data is established with voltage levels of 0 volts for “active state” and 5 volts for “idle state”.

Voltages under 1 volt are considered as “active” while voltages over 3,5 volts are considered as “idle”.

## 1.2 COMMUNICATION FEATURES

The communication is asynchronous and half-duplex, that is, more than one element cannot be transmitted on the bus at the same time. 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, such as

| | | | | |
|---|---|---|---|---|
| **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 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.

AZKOYEN
MEDIOS DE PAGO

### 1.2.1 Destination address

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

| Address | Description |
|---|---|
| **0** | Used in messages which affect all the slave devices simultaneously *(Broadcast messages).* |
| **1** | Machine address. |
| **2** | Address of the device in communication with only one slave (habitually reserved for the validator). |
| **3 - 255** | Addresses of the devices in multi-slave communication. The default address of the hopper T3 is 3. |

### 1.2.2 Number of data bytes

The range of the number of data to be transmitted is from 0 to 252 0 a 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 Hopper addresses).

| Address | Description |
|---|---|
| **1** | Machine address. |
| **2** | Address of the device in communication with a single slave (habitually reserved for the validator). |
| **3 - 255** | Addresses of the devices in multi-slave communication. The default address of the hopper T3 is 3. |

### 1.2.4 Header

The range of the header byte is from 0 to 255.

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 hoppers, the header will have the value’0’ in all 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.

***For 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

### 1.3.1 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.

### 1.3.2 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.

### 1.3.3 Maximum time from power up until 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 200 ms.

## 1.4 ERROR MANAGEMENT

### 1.4.1 Action taken in the case of error

If a Hopper receives an incomplete message (reception timeout) or with an incorrect checksum, the only action is carries 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.

**It is recommended to do various retries if, at moment the Hopper is paying out, the reply to a command from the machine is not received**

The recommended waiting time for sending the retry 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-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 or emptying command is received.

### 1.4.2 Error Communication

The Hopper 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 of Status” command requested by the latter from 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.

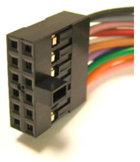
The errors are cumulative, that is, if more than one error is being detected at the same time, the Machine will be notified of all of them.

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.

# 2 CCTALK PROTOCOL IN THE HOPPER T3

## 2.1 CCTALK CONNECTORS

***Azkoyen type***

| |  |
|---|---|
| **Azkoyen connector in the base**<br>**Molex 2x6 female 2.54** | **Recommended for the master**<br>**Molex 2x6 male 2.54** |

***Cinch Type***

| | |
|---|---|
| **Cinch connector female 12-way** | **Cinch connector male 12-way** |

***Pinout connector ccTalk (for both types)***

| Pin | Function |
|---|---|
| 1 | 0V (GND) |
| 2 | NC |
| 3 | NC |
| 4 | Val. Address 1 (LSB) |
| 5 | DATA (Cctalk) |
| 6 | NC |
| 7 | NC |
| 8 | Val. Address 2 |
| 9 | +Vs (+24V) |
| 10 | NC |
| 11 | NC |
| 12 | Val. Address 3 ( MSB) |

## 2.2 CONFIGURATION OF THE DIPSWITCHES

The Hopper has a Dipswitch of 8 configuration positions.

| Dipswitch No. | Function |
|---|---|
| **SW1 - SW4** | Selection of the ccTalk address (see details below) |
| **SW5** | **OFF** – External addressing (via connector)<br>**ON** – Internal addressing (dipswitches) |
| **SW6** | No used. Reserved for future use. |
| **SW7** | No used. Reserved for future use. |
| **SW8** | **OFF** – Standard payout<br>**ON** – Encrypted payout |

## 2.3 CCTALK ADDRESSING

The ccTalk address of the Hopper T3 can be configured by:

- Addressing with the connector.
- Addressing with the dipswitch.
- Addressing with the CCTALK commands.

If **SW5** is **OFF**, the address is taken from the connector.
If **SW5** is **ON**, the address is taken from the dipswitches.

The address of the device can be modified at any time with CCTALK MDCES commands. This method of addressing is saved in the volatile memory so it will be lost after each shutdown, and will then be taken from SW5 (connector or dipswitches) on the next start-up.

### 2.3.1 ADDRESSING BY CONNECTOR

This addressing is done with hardware, using the three address lines on the main connector. The hopper has a default address 3 assigned on the CcTalk bus, as a payout device. In applications where more than one hopper is used, it is necessary to configure the addresses for the rest of the devices, connecting one or various address selection lines to the power +Vs on the connector on the machine.
SW5 should be OFF to address with this method.

***Configuration of the addresses using hardware***

| *CcTalk address* | *Sel. Add.1* | *Sel. Add.2* | *Sel. Add.3* |
|---|---|---|---|
| *3* | … | … | … |
| *4* | ● | … | … |
| *5* | … | ● | … |
| *6* | ● | ● | … |
| *7* | … | … | ● |
| *8* | ● | … | ● |
| *9* | … | ● | ● |
| *10* | ● | ● | ● |

**Note:**

| | |
|---|---|
| **…** | **Not connected or GND** |
| ● | **Vx between 5V and 24V** |

## 2.3.2 ADDRESSING WITH SWITCHES

The ccTalk address is established by configuring the dipswitches on the control board of the Hopper T3 as shown in the table below. This addressing is effective when SW5 is ON.

***Table of the addressing with dipswitches***

| *ccTalk Address* | *SW1* | *SW2* | *SW3* | *SW4* |
|---|---|---|---|---|
| *3* | OFF | OFF | OFF | OFF |
| *4* | ON | OFF | OFF | OFF |
| *5* | OFF | ON | OFF | OFF |
| *6* | ON | ON | OFF | OFF |
| ………………… | | | | |
| *18* | ON | ON | ON | ON |

## 2.3.2 ADDRESSING WITH CCTALK COMMANDS

Addressing can be carried out using CcTalk commands, **Address change [251]** and **Address random [250]**. For more information, consult the specific command below.

# 3 BASIC START-UP

## 3.1 PAY-OUT SEQUENCE

The following is a basic sequence for carrying out a pay-out:

**i.** First activate the Hopper using the Command **Enable Hopper [164]**:

| **COMMAND** (**Master** → Hopper) | **REPLY** (Master ← **Hopper**) |
|---|---|
| **ENABLE HOPPER**<br>[Destination] [1] [Origin] [Command 164] [165] [CHK] | |
| | **ACK**<br>[ Destination ] [ 0 ] [ Origin ] [ 00 ] [ CHK ] |

**ii.** After that, the machine requests a valid **Cipher Key** to encrypt the payment:
(This step is not necessary if the Hopper is not encrypted)

| **COMMAND** (**Master** → Hopper) | **REPLY** (Master ← **Hopper**) |
|---|---|
| **CIPHER KEY**<br>[Destination] [0][Origin][Command 160] [CHK] | |
| | [Destination] [8][Origin][00] [Key 1] [Key 2] [Key 3] [Key 4] [Key 5] [Key 6] [Key 7] [Key 8] [CHK] |

**iii.** The machine will send the **Payment Order** with the 8 correct digits (whether it is encrypted or not), and the number of coins to pay out.

| **COMMAND** (**Master** → Hopper) | **REPLY** (Master ← **Hopper**) |
|---|---|
| **DISPENSE HOPPER COINS**<br>[Destination] [9] [Origin] [Command 167] [Data 1] [Data 2] [Data 3] [Data 4] [Data 5] [Data 6] [Data 7] [Data 8]  [Nº Coins to pay] [CHK] | |
| | [Destination] [1][Origin][00] [Nº of Payment] [CHK] |

**iv.** Once the Hopper starts the payout, the machine checks the process using the Command 166 Request Hopper Status, until the coins left to pay out is 0.

| **COMMAND** (**Master** → Hopper) | **REPLY** (Master ← **Hopper**) |
|---|---|
| **REQUEST HOPPER STATUS**<br>[Destination] [0] [Origin] [Command 166] [CHK] | |
| | [Destination] [4] [Origin] [0] [Nº of payment] **[To be paid]** [Paid] [0] [CHK] |

## 3.2 POST PAYMENT

After each payment, you should check the status of the Hopper, the coin levels, the status of the last payout and carry out a Test to detect possible errors, therefore we recommend:

- **Send Command 166 (Request hopper status):**
  Check the payout finished correctly.

- **Send Command 163 (Test hopper):**
  Check there are no errors and that the Hopper is working correctly.

- **Send Command 217 (Request Payout High / Low Status):**
  Check the coin levels are adequate.

# 4 USE OF ENCRYPTING IN COMMUNICATION

The Hopper T3 is equipped with an encryption system CMF. If increased payout security is required, it can be activated using switch SW8.

Once the Hopper is configured for encryption, the payment command will be protected with encryption, so to achieve a correct payout we should follow the sequence below:

**i.** First, the machine should request the Cipher key [8 bytes]. The Hopper will randomly generate one and save it. It will be sent to the machine with the following sequence:

| **COMMAND** (**Master** → Hopper) | **REPLY** (Master ← **Hopper**) |
|---|---|
| CIPHER KEY<br>[Destination] [N º Data][Origin][Command A0h] [CHK] | |
| | [Destination] [N º Data][Origin][Command 00] [Key 1] [Key 2] [Key 3] [Key 4] [Key 5] [Key 6] [Key 7] [Key 8] [Checksum] |

Then the payout order is carried out.

**ii.** The machine will send the encrypted data together with **the number of coins to pay out:**

| **COMMAND** (**Master** → Hopper) | **REPLY** (Master ← **Hopper**) |
|---|---|
| **PAYMENT ENCRYPTED**<br>[Destination] [N º Data][Origin][Command A7h] [Data 1] [Data 2] [Data 3] [Data 4] [Data 5] [Data 6] [Data 7] [Data 8] [Nº Coins to pay] [Checksum] | |

With this data and the cipher key, the Hopper will decode the message will respond with the payout and will carry out the payment.

Before this sequence, you could send the Command 161 (Pump RNG) with random data to increase the randomness of the Keys.

Each time a payment is made, the Hopper will cancel the cipher key used. This obliges the machine to request a new cipher key for the next payment.

# 5 PROTECTION WITH PIN

The protection with PIN is a basic security based on a 4-byte PIN (Between 0 and 4,294,967,295). The Hopper T3 leave the factory with the default PIN “00-00-00-00”. While the Hopper has this PIN, the protection is deactivated.

This security is activated when a new PIN other than 0 with the Command 219 (Enter New PIN number) is programmed. After programming the new PIN the Hopper will not need the PIN to be entered again until the hopper has been reset.

If the Hopper is protected, it will impede the introduction of another new PIN or any payout. To allow these commands, we need to introduce the correct PIN after each reset, using the command 218 (Enter PIN number).

To deactivate this Protection, programme the new PIN to “00-00-00-00”.

This Protection requires the knowledge of the programmed PIN; otherwise the Hopper will be unable to make a payout. For this reason great care must be taken with the use of the PIN. Below is some advice on the use of the PIN:

- Always use the same PIN for an exploitation or client, etc…
- Store the Pin in an encrypted code in the memory bank of the Hopper so the machine can recuperate it easily.
- Maintain a data base of the Serial Numbers and PIN Numbers.
- Force the reset to “00-00-00-00” before disconnecting the Hopper from the machine.

# 6 TREATMENT OF EVENTS

## What happens after a power-up or a Software Reset?

The following is a description of what happens after each reset and power-up:

- **CCTALK ADDRESSING:**
  That indicated with SW5 (Connector or Switches)

- **PIN NUMBER:**
  If it is activated, it will have to be introduced.

- **PAYMENT VARIABLES:**

| | |
|---|---|
| [*LimitOfCurrent*] | (Default: 20 is equivalent to 1.2 A) |
| [*RetardStopMotor*] | (Default: 0 ms) |
| [*MaximumSpan*] | (Default: 60 is equivalent to 20 sec.) |
| [*I.Max*] | 0 A |
| [*Pay1By1*] | Payout Standard (Multiple) |

- **FLAGS:**
  All Flags are reset, except the Power-Up which appears after Power up and is reset to 0 after the Software Reset.
  Also the Hopper is deactivated as protected.

- **COUNTERS:**

| | |
|---|---|
| [*HopperDispenseCount*]: | The value previously stored is maintained. |
| [*HopperLifeDispenseCount*]: | The value previously stored is maintained. |

- **REQUEST HOPPER STATUS:**

| | |
|---|---|
| [*CoinsToBePaid*]: | Is reset to 0. |
| [*CoinsPaidLastPayout*] | The value previously stored is maintained. |
| [*CoinsToBePaidLastPayout*] | The value previously stored is maintained. |

- **VARIABLES REQUEST COMMS STATUS:**

| | |
|---|---|
| [*RX Timeouts*]: | Is reset to 0. |
| [*BytesIgnored*] : | Is reset to 0. |
| [*ChecksumsErroneous*]: | Is reset to 0. |

AZKOYEN
MEDIOS DE PAGO

# 7 LIST OF COMMANDS

### Command 254: Simple poll

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

| | |
|---|---|
| *Data sent:* | -- |
| *Data received:* | **ACK** |

### Command 253: Address poll

This command requests all slave devices for their address. The command is sent with the destination address of 0 (broadcast). To avoid collisions, the slave devices do not reply with a standard cctalk message, but reply with the byte of the address with a delay that is proportional to the value of the address.

| | |
|---|---|
| *Message sent:* | **[00] [00] [01] [FDH] [02]** |
| Message received: | **{Delay variable} [Dir]** |

<u>Where:</u>

Dir = address of the corresponding Hopper.

The algorithm used to calculate the delay of the reply is the following:

```
Deactivate rx port
Delay (4*Dir) milliseconds
Send [Dir]
Delay 1200  (4*Dir) milliseconds
Activate rx port
```

### Command 252: Address clash

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

To avoid collisions, the slave devices do not reply with a standard cctalk message, but reply with the byte of the address with a byte is only returned with a random delay in the sending.

| | |
|---|---|
| *Message sent:* | **[Dir] [00] [01] [FCH] [Chk]** |
| *Message received:* | **{Delay variable} [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)
Deactivate rx port
Delay (4*Dir) milliseconds
Send [Dir]
Delay 1200  (4*Dir) milliseconds
Activate rx port
```

**Command 251: Address change**

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 dip switch, 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 to find out the new address of the Hopper.

| | |
|---|---|
| *Data sent:* | **[Dir]** |
| *Data received:* | **ACK** |

Where:

Dir = new address of the corresponding Hopper

The new address is maintained after switching off or a reset.

When SW6 is ON, the Hopper will not carry out any action for this command (it will not even respond).

## Command 250: Address random

This command permits reprogramming the Hopper with a new address whose value will be random and generated by itself.

This is an escape channel for cases when several devices share the same address.

After this command the machine must send the Address Poll command to know the values of the new addresses.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **ACK** |

The new address is maintained after switching off or a reset.
When SW6 is ON, the Hopper will not carry out any action for this command (it will not even respond.

## Command 247: Request variable set.

Using this command the Hopper T3 informs the machine on how it is configured.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[LimitOfCurrent] [RetardStopMotor] [MaximumSpan] [I.Max][Vdc][DirectionConnector]** |

<u>Where:</u>

| | |
|---|---|
| **[LimitOfCurrent]** | Limit of the current where it considers necessary the start of a jam clearing procedure. (Default: 20 is equivalent to 1.2 A) |
| **[RetardStopMotor]** | Not used (Default: 0 ms) |
| **[MaximumSpan]** | Programme the Maximum time span of no coin payout between coins. (Default: 60 is equivalent to 20 sec.) |

| | |
|---|---|
| **[I.Max]** | Indicates the maximum current measured by the Hopper (Data/16.6 = Amp). |
| **[Vdc]** | Indicates the Value of the voltage. (0.2+Data*0.127 = Volts) |
| **[DirectionConnector]** | Indicates the direction forced externally |

The equivalency of the data to magnitudes can be seen in table X.

### Command 246: Request manufacturer ID

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

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **"Azkoyen"** |

### Command 245: Request equipment category ID

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

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **"Payout"** |

### Command 244: Request product code

With this command, the hopper gives the machine the product code of the device.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **DATA** |

Where:

DATA= "SUH1noEnc" it is configured as Not Encrypted.

DATA= "SUH1Enc" it is configured as Encrypted.

**Command 242: Request serial number**

With this command, the corresponding Hopper sends the machine the serial number of the device with a code of 3 bytes. This command has been implemented so the machine has a security code which is necessary for the payout of the coins.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[Serial 1 - LSB] [Serial 2] [Serial 3 - MSB]** |

The serial number is unique for each Hopper manufactured.

**Command 241: Request software version**

With this command, the corresponding Hopper sends the Machine the current software version of the device. Cctalk® does not establish any restriction concerning the format of the reply message.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | “**V1.0"** |

**Command 236: Read opto states**

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

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[Data 1]** |

Where:

Data 1 = 0x00 Detector clear.

Data 1 = 0x87 Detector covered.

**Command 219: Enter New PIN number**

This command is used to change, as often as required, the PIN number of the Hopper. As it is a command that protected with a PIN, the present PIN number must be used to be able to use the command.

| | |
|---|---|
| *Data sent:* | [PIN 1] [PIN 2] [PIN 3] [PIN 4] |
| *Data received:* | **ACK** |

The new PIN number will be maintained after power off or a reset.

Modifying the PIN number to 0, will deactivate the protection with PIN numbers.

## Command 218: Enter PIN number

The PIN number can be introduced with this command. The PIN number is a binary code of 32 bits (4.294.967.296 combinations).

| | |
|---|---|
| *Data sent:* | [PIN 1] [PIN 2] [PIN 3] [PIN 4] |
| *Data received:* | **ACK** |

- If the PIN number introduced is correct or incorrect, the Hopper will send an (ACK).
- If the PIN system is active (PIN Number different to 0), all commands implemented in the Hopper, except the command “ADDRESS POLL, 253” are protected.
- If the PIN is activated and the command does not have the correct PIN, there will be no answer from the Hopper.
- If the PIN is activated, it must be sent after every power off and reset so that the Hopper will execute the command.

## Command 217: Request Payout High / Low Status

Using this command, the hopper is consulted about the status of the full and empty sensors.

| | |
|---|---|
| *Data Sent:* | --- |
| *Data received:* | [Byte] |

**Bit 0 –** Status of the empty sensor.

0 = Number of coins greater or equal to the empty level

1 = Number of coins less than the empty level

**Bit 1 –** Status of the full sensor

0 = Number of coins less than the full level

1 = Number of coins greater or equal to the full level

**Bit 2 - <Reserved>**

**Bit 3 - <Reserved>**

**Bit 4 – Detection of the Empty Sensor**

0 = Not connected

1 = Connected

**Bit 5 – Detection of the Full Sensor**

0 = Not connected

1 = Connected

**Bit 6 - <Reserved>**

**Bit 7 - <Reserved>**

**Command 216: Request data storage availability**

The Hopper T3 has an area in the memory so the machine can use it for different functions. Using this command, the hopper informs the machine on the characteristics of this memory.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[MemoryType] [ReadBlocks] [ReadBytesPerBlock] [WriteBlocks] [WriteBytesperBlock]** |

**The map of this memory is the following:**

| Block No. | Bytes Length | Description | Read / Write |
|---|---|---|---|
| 0 | 8 | User data | R / W |
| 1 | 6 | Coin name | R / W |
| 1 | 2 | User data | R / W |
| 2 | 3 | Hopper dispense count | R / W |
| 2 | 1 | Checksum A | R / W |
| 2 | 1 | Last payout: coins paid | R / W |
| 2 | 1 | Checksum B | R / W |
| 2 | 1 | Last payout: coins unpaid | R / W |
| 2 | 1 | Checksum C | R / W |
| 3 | 3 | Hopper life dispense count | Read |
| 3 | 1 | Checksum D | Read |
| 3 | 1 | Black Box Recorder A | Read |
| 3 | 1 | Black Box Recorder B | Read |
| 3 | 1 | Black Box Recorder C | Read |
| 3 | 1 | Black Box Recorder D | Read |

**Command 215: Read data block**

With this command the machine can read any memory block in the hopper.

| | |
|---|---|
| *Data sent:* | **[BlockNumber]** |
| *Data received:* | **[Data1]…[Data8]** |

Where:

| | |
|---|---|
| **[BlockNumber]** | Nº of the memory block to be read. |
| **[Data1]…[Data8]** | Data stored in the Hopper. |

**Command 214: Write data block**

With this command the machine can manipulate the blocks that allow writing, to reset counters or store data.

| | |
|---|---|
| *Data sent:* | **[BlockNumber] [Data1]…[Data8]** |
| *Data received:* | **ACK** |

Where:

| | |
|---|---|
| **[BlockNumber]** | Nº de memory block that needs rewriting. |
| **[Data1]…[Data8]** | Data to be stored in the Hopper. |

**Command 192: Request build code**

With this command, the corresponding Hopper sends the Machine the device build code.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **"T3_PLUS1"** |

**Command 172: Emergency stop**

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

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[Data 1]** |

Where:

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

Also, if the Hopper is paying out, after responding with the ACK the Hopper will reset.

**Command 171: Request hopper coin**

With this command the Hopper T3 responds with an ASCII code with the name of the configured currency. To read this information, the machine must have previously had to write it using the Command 214 (Write data block).

If not, the Hopper responds with the default 6x ASCII code 0

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **6x ASCII Code** |

**Command 169: Request address mode**

With this command the Hopper informs the machine on the methods it has for its CCTALK addressing.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[AddressMode]** |

Where:

Bit0 - Address is stored in ROM
Bit1 - Address is stored in RAM
Bit2 - Address is stored in EEPROM or battery-backed RAM
Bit3 - Address selection via interface connector
Bit4 - Address selection via PCB links
Bit5 - Address selection via switch
Bit6 - Address may be changed with serial commands (volatile)
Bit7 - Address may be changed with serial commands (non-volatile)

**6A** hex = Address stored in RAM
Address selection via interface connector
Address selection via switch
Address may be changed with serial commands (volatile)

**Command 168: Request hopper dispense count**

This command causes the Hopper to send the Machine the total number of coins paid since the last reset. This counter is stored in non-volatile memory and can be reset using the command **Write Data Block.**

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[Data 1] [Data 2] [Data 3]** |

Where:

[Data 1] = Number of coins paid (LSB).
[Data 2] = Number of coins paid.
[Data 3] = Number of coins paid (MSB).

Number of coins paid is between 0 and 16,777,215.

**Command 167: Dispense Hopper coins**

With this command, the machine orders the Hopper to execute the different payments of coins, from 1 to 255 coins. This command can be protected with encrypted 64Bit protection. If the encryption is not correct, the Hopper will not make the payout.

| | |
|---|---|
| Data sent: | **[Key1] [Key2] [Key3] [Key4]** |
| | **[Key5] [Key6] [Key7] [Key8] [Nº coins to pay]** |
| *Data received:* | **[Nº de Pago]** |

If the Hopper is not encrypted, any 8-bit code can be. For example:

[00] [00] [00] [00] [00] [00] [00] [00] [Nº Coins].

If the Hopper can execute the command, it will increment the payout number and will immediately start payment until one of the following events:

- Extraction of all the coins indicated
- Detection of a maximum span
- Reception  of a cancel command
- Detection of an error
- Reset hardware (power failure) and software.

In the event that the Hopper cannot execute the command, it returns a negative acknowledgement string and continues in the status it was in.  Sending this negative acknowledgement may be due to the following reasons:

- Hopper in error state
- Safety code received not correct

If the hopper has started a payment sequence and a new one is requested, the hopper will not respond.

The payment number is stored in the volatile memory, so it will be reset after any Reset.

## Command 166: Request hopper status

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

|  |  |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[Data 1] [Data 2] [Data 3] [Data 4]** |

Where:

If the Hopper is paying out when command is received:

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

[Data 2] = Number of coins still to be paid

[Data 3] = Number of coins paid in last payment

[Data 4] = 0

If the Hopper is at rest when this command is sent:

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

[Data 2] = 0

[Data 3] = Number of coins paid in last payment

[Data 4] = Number of coins still to be paid for last payment

When the number of payouts (Data 1) is 255, the next payment command sets the counter to 1. The only way that the counter will go to 0, is after a reset.

The information on the coins paid out and still to be paid is stored in the non-volatile memory so it can therefore be recuperated after a power loss and the payment can be continued on the next power up.

## Command 165: Modify variable set

Using this command the machine configures some specific functions in the Hopper T3.

|  |  |
|---|---|
| *Data sent:* | **[LimitOfCurrent] [RetardStopMotor] [MaximumSpan] [PayOf1By1]** |
| *Data received:* | **ACK** |

Where:

|  |  |
|---|---|
| **[LimitOfCurrent]** | Limit of the current for starting a jam clearing routine. |

(Between 6 and 30: between 0.35A and 1.8A)

| | |
|---|---|
| **[RetardStopMotor]** | Not used.(Programmed 0 msec in factory) |
| **[MaximumSpan]** | Programme a maximum time span between payment of coins. (Not less than 30: 10sec) |
| **[PayOf1By1]** | If it is 1, payment 1 by 1 will be permitted. (0 or other than 1 actives Standard payout) |

**Command 164: Enable hopper**

This command should be used to activate the Hopper before using any payout command.

| | |
|---|---|
| *Data sent:* | **[Data 1]** |
| *Data received:* | **ACK** |

Where:

If Data1 = A5H, the hopper activates

If Data 1 is not A5H the hopper deactivates.

**Command 163: Test hopper**

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.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[Data 1] [Data 2]** |

Where:

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 - Exit Sensor - coin blocked at rest.

Bit 4 – Detected coin exit at rest.

Bit 5 – Opto blocked during payment.

Bit 6 – Power Up Detected
Bit 7 - Payment disabled.

Data 2 =

Bit 8 - Fault in counting sensor or external light.
Bit 9 – 1 coin method of payout
Bit 10 – Checksum A error.
Bit 11 – Checksum B error.
Bit 12 – Checksum C error.
Bit 13 – Checksum D error.
Bit 14 – Not detected movement of chain during payout.
(Reducer gear, Encoder or Slot Sensors).
Bit 15 – Protection of Pin Activated

**1 = True, 0 = False**

**Command 161: Pump RNG**

With this command the machine sends 8 random data to the hopper. This new data helps the Hopper to increase the randomness of its Codes.

| | |
|---|---|
| *Data sent:* | **[random 1] [random 2] [random 3] [random 4]** |
| | **[random 5] [random 6] [random 7] [random 8]** |
| *Data received:* | **ACK** |

**Command 160: Request cipher Key**

| | |
|---|---|
| *Data sent:* | **---** |
| *Data received:* | **<Variable>** |

This command requests the slave device for a password that forms part of the encrypting algorithm of the payment system**.**

**Command 141: Request firmware upgrade capability.**

| | |
|---|---|
| *Data sent:* | **---** |
| *Data received:* | **[firmware options]** |

Where:

firmware options=

0 if the firmware is in ROM /EPROM

1 if the firmware is in FLASH/EPROM with update capacity.

**Command 140: Upload firmware**

This command is used to update the firmware of the device.

| | |
|---|---|
| *Data sent:* | **[block] [line] [data1] [data2] ... [data128]** |
| *Data received:* | **ACK** |

**Command 139: Begin firmware upgrade**

| | |
|---|---|
| *Data sent:* | **---** |
| *Data received:* | **ACK** |

When the Hopper receives this command, the firmware updating commences.
This command is not available for the Italian market, they cannot be updated using cctalk commands, and the reply will be a "**NACK**" to this command.

**Command 138: Finish firmware upgrade**

| | |
|---|---|
| *Data sent:* | **---** |
| *Data received:* | **ACK** |

When the Hopper receives this command, the firmware updating finishes.

**Command 04: Request comms revision**

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.

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **[cctalk Level] [major revision] [minor revision]** |

Where:

cctalk Level = 01H
Major revision= 04H
Minor revision= 03H

**Command 03: Clear comms status variables**

This command Resets the counters reported by the command **Request comms status variables**

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **ACK** |

**Command 02: Request comms status variables**

Thanks to this command, you can detect some of the most common problems with the serial communication. Using this command, the Hopper informs on the different problems occurring.

| | |
|---|---|
| *Data sent:* | **[ Timeouts] [BytesIgnored] [ChecksumsErroneous]** |
| *Data received:* | **ACK** |

Where:

[RX Timeouts]: Counter of Timeouts between bytes occurred.
[BytesIgnored]: No used, default = 0.
[ChecksumsErroneous]: Counter of Checksums Erroneous occurred.

**Command 01: Reset device**

| | |
|---|---|
| *Data sent:* | --- |
| *Data received:* | **ACK** |

This command causes the Hopper to carry out a software reset. The affected Hopper sends a positive acknowledgement string immediately before making the reset.

This command resets the Power up Bit as well as resetting the previously mentioned registers.

When a Reset command is sent to the Hopper, it is necessary to wait at least 50 milliseconds before sending a Payment command A7.