

# 1. AZKOYEN protocol (serial)

In this operating mode, both the payout unit / machine communications (coin output detection, empty, full, scale...) as well as machine / payout unit communications (payment of a certain number of coins, empty command, status request...) take place through an asynchronous serial communications bus at TTL levels, thereby using the communications protocol described in this chapter.

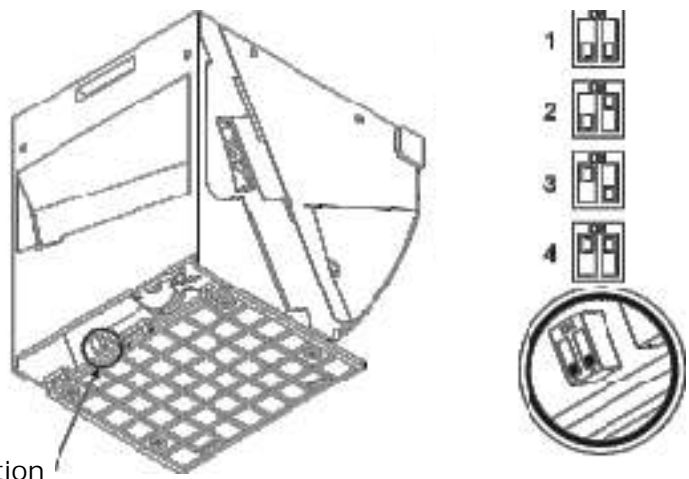
This communications bus allows connecting up to 4 payout units using a single, 10-pin flat cable to which the connectors of these hoppers are spliced in, thereby simplifying the cabling considerably and decreasing the cost of the same.

## 1.1. Address configuration

The addresses of the modes are configured using a 2-way dip switch that can be accessed from the underneath the payout unit, as shown in the following illustration

This dipswitch allows 4 different combinations, as shown in the following table:

Sw1	Sw2	
OFF	OFF	1
OFF	ON	2
ON	OFF	3
ON	ON	4



Configuration

## 1.2. Signals

The Hopper "U" controls coin output using an optical sensor that detects the coin passing through an output channel.

### Errors

The Hopper "U" has the capacity to detect the following errors:

Error N.	Code	Description	Meaning
Error 1	01H	Permanent output detection	The output of a coin has been detected during a time greater than the estimated time
Error 2	02H	Coin output detection in stand-by mode	The output of a coin with the payout unit stopped has been detected
Error 3	04H	Permanent jam detection	The payout unit could not clear the jam and the maximum time has been exceeded
Error 4	08H	Motor not detected	The motor did not start when attempted

### Jam

A power detector is used to detect the moment when the Hopper jams, and at that moment it reverses the rotation direction for 1 second in order to clear the jam.

### Span

If a span exceeding 5 seconds is detected, the Hopper reverses the motor rotation direction for 1.5 seconds in order to move the coins around. In operating Mode 3, if a certain number of consecutive spans occurs, coin extraction ends, and the Hopper subsequently communicates Error 4 to the machine.

### Empty and Full (for hoppers that include them)

Emptying and filling are detected by means of an optical detector consisting of a photodiode and a phototransistor.

### Scale (for hoppers that include one)

A microswitch contact located on the mechanical scale detects that the level of coins stored in the hopper has reached the maximum value for which it has been adjusted.

## 1.3. Development of the AZKOYEN protocol

### Communications topology

Depending on the reference number of the card that the payout unit contains and with which it is working, communication can be by two wires (TxD/RxD) or by one wire (DATA). The data are established by TTL levels: 5 volts is a "0" logic and 0 volts is a "1" logic, and the stand-by level is 5 volts.

### Communication characteristics

Communication is asynchronous and half-duplex, meaning that more than one bus element cannot be simultaneously transmitted.

Communication timing complies with the characteristics of the standard industrial RS-232.

RS-232 communication has various parameters that are configured in this application as follows:

4800 or 9600 baud rate, 1 start bit, 8 data bits, without parity, 1 stop bit.

### Message structure

Each communication sequence is formed by two frames. The first corresponds to the command sent by the Machine to the Payout Unit, and the second is the response sent by the Payout Unit to the Machine. Both frames have the format indicated in the box.

```
[Destination address]
[Number of data bytes]
[Origin address]
[Header]
[Data 1]
[Data 2]...
[Data N]
[Checksum]
```

### Destination address

The range of addresses goes from 0 to 255 (254 of them correspond to Payout Unit addresses, as explained below).

- 0: Used in messages that affect all Payout Units simultaneously, although in the current version there is no implemented command that uses it.
- 1: Address of the Machine.
- 2: Address of the Payout unit in communications with a single Payout Unit.
- 3 to 255: Addresses of the Payout Units in multi-payout unit communications. The "destination address" byte indicates the node of the bus where the message is directed. In this case, the values that are used for the destination address are the following:

1: When the message is directed from a payout unit to a machine  
3, 4, 5 or 6: When the message is directed to the payout unit configured using the dipswitch as 1, 2, 3 or 4, respectively.

### 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," it means that the message has no data, and in this case the total number of message bytes will be 5 bytes (the minimum allowed).

The values 253 to 255 are not allowed in this field, and they would be considered as the value 252.

### Origin address

The range of addresses goes from 1 to 255 (254 of them correspond to payout unit addresses).

- 1: Address of the Machine.
- 2: Address of the Payout unit in communications with a single Payout Unit.
- 3 to 255: Addresses of the Payout Units in multi-payout unit communications, which is this case. The "origin address" byte indicates the node of the bus that is sending the message. In this case, the values that are used for the "origin address" are the following:

1: When the message is sent by the machine  
3, 4, 5 or 6: When the message is directed to the payout unit configured using the dipswitch as 1, 2, 3 or 4, respectively.

### Header

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

The "header" will never have a value of "0" in messages sent by the machine.

As regards response messages, meaning those sent by the payout units, the "Header" will have a "0" value in all messages, except in "Negative acknowledgement" or NACK messages.

## Data

The range of values that each one of the data bytes can have is from 0 to 255. It has no restrictions on use, and any format can be used, such as binary, ASCII...

## Checksum

The "Checksum" that which makes the 8 bits of least weight of the sum of all bytes of the message, including the checksum itself, equal "0."

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

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

## Time requirements

### Maximum time between bytes

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

### Maximum time between command and response

The maximum response time to a command depends on the time that it takes the payout unit to process that command. This time is 50 milliseconds by default, if no other value is specified in the command definition.

### Minimum time from supplying power until the first command is sent.

The minimum time that has to elapse from the moment that the payout unit is powered until it is sent the first command must be 100 milliseconds.

## Error management

### Actions in the event of an error

If a payout unit receives an incomplete message (reception timeout) or one with an incorrect checksum, it takes no action other than to initialise the communications variables and prepare to receive a new message. The machine, since it does not receive any response to the message sent, can choose to resend the same message.

Conversely, if the machine receives an incomplete message (reception timeout) or one with an incorrect checksum, it can choose to resend the same message. In any event, when there is an error upon receiving a message, there is no negative acknowledgement message defined, which simplifies the implementation of multi-payout unit protocols and reduces the collisions.

If a payout unit receives a command that it cannot execute, it responds with a negative acknowledgement message. This occurs, for example, when another pay out or empty command is received during the execution of a pay out command.

Error communication

The payout unit performs a series of controls of its peripheral devices, and it is capable of detecting a series of anomalies, which are subsequently communicated to the machine.

The errors that occur are reported to the machine by means of the "Status Request" command requested by the machine to the payout unit. The payout unit responds to the request with a byte that indicates its error status and with another byte that indicates the error that has occurred. Each one of the bits of this latter byte represents a possible error, such that the bits that are at 1 will indicate that the corresponding error has been detected, and those that are at 0 will indicate that they have not been detected.

The errors are accumulative, meaning that if more than one error is being detected at the same time, they are all reported to the machine.

The payout unit exits the error status when it receives a "Pay out," "Empty" or "Reset" command, thereby executing this command. If the failure persists, the payout unit will again enter the error status.

*Among the commands implemented, there are some that are specific to cctalk® version 2.0 and others of the Azkoyen protocol.*

Commands

Code	Command	Data bytes TRANSMISSION	Data bytes RESPONSE
<b>Cctalk® Commands</b>			
FEH	Simple Poll	0	0
F5H	Request ID	0	6
F1H	Request Software Revision	0	2
04H	Request Comms Revision	0	3
ECH	Read opto states	0	1
01H	Reset Device	0	-
<b>Azkoyen protocol commands</b>			
10H	Estado Ultimo Comando	0	De 1 a 5
11H	A cerca de...	0	5
12H	Estado	0	De 1 a 5 (Ver Tabla 2)
13H	Cancelación	0	0
14H	Vaciado	0	0
15H	Pago	2	0

Simple Poll [FEH = 254d]

Command for verifying the correct operation of communications and for confirming the presence of a specific payout unit on the bus.

Transmission: [Dir][00][01][FEH][Chk]  
 Response: [01][00][Dir][00][Chk]

In the event that no response is received to the request sent (Reception timeout on the machine), it is a signal that the corresponding payout unit is impaired or not connected.

**Example:**

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

Transmission: [04][00][01][FEH][FDH]  
 Response: [01][00][04][00][FBH]

The machine verifies that the payout unit coded as number 3 on the bus is NOT present.

Transmission: [05][00][01][FEH][FCH]  
 Response: The hopper does not respond

## Request equipment category ID [F5H = 245d]

This command allows receiving from the corresponding payout unit the chain of characters that identifies the kind of device in question. As regards coin payout units, which is this case, "Payout" is received.

Transmission: [Dir][00][01][F5H][Chk]  
 Response: [01][06][Dir][00][Data 1][Data 2][...][Data 6][Chk]  
 Where: Data 1 Data 6 = character chain  
 Dir = address of the corresponding payout unit

**Example:**

The machine requests from payout unit number 1 the character chain that indicates the kind of device in question.

Transmission: [03][00][01][F5H][07H]  
 Response: [01][06][03][00][50H][61H][79H][6FH][75H][74H][74H]  
 Note: The bytes [50H], [61H], [79H], [6FH], [75H] and [74H] correspond to the characters "P," "a," "y," "o," "u" and "t," respectively.

## Request for software version [F1H = 241d]

With this command, the corresponding payout unit sends the current software version to the machine. Cctalk® does not set any restriction as regards the format of the response message.

Transmission: [Dir][00][01][F1H][Chk]  
 Response: [01][02][Dir][00][Data 1][Data 2][Chk]  
 Where: Data 1 = Byte before the decimal point of the program version  
 Data 2 = Byte after the decimal point of the program version  
 Dir = Address of the corresponding payout unit

**Example:**

The machine requests from payout unit number 3 its software version, which in this example is version 9.2.

Transmission: [05][00][01][F1H][09]  
 Response: [01][02][05][00][09][02][EDH]  
 Note: The payout unit sends to the machine that the software version is 9.2.

Read opto states [ECH = 236d]

The payout unit responds to this command by sending one byte, therein indicating the value of the empty and full detection optics and detection of the mechanical scale.

Transmission: [Dir][00][01][ECH][Chk]  
 Response: [01][01][Dir][00][Data 1][Chk]  
 Where: Data 1 = Value of the detectors  
 Data 1.bit0 = Empty detector  
 Data 1.bit1 = Full detector  
 Data 1.bit2 = Scale detector  
 Data 1.bit3 = Maximum detector  
 Dir = Address of the corresponding payout unit

The table reflects the value of these empty and full bits (bits 0 and 1) according to the coin load in the payout unit.

Load	bit 0	bit 1
Empty	1	1
½ load	0	1
Full	0	0

Example:

The machine requests from payout number 2 the values of the empty, full and scale detectors.

Transmission: [04][00][01][ECH][Chk]  
 Response: [01][01][04][00][05][F5H]  
 Note: It is deduced from this response that the empty detector = 1, the full detector = 0 and the scale detector = 1.

Request comms version [04H]

As the response to this command, the payout unit sends the implementation level of the cctalk® protocol (01H in this case) and the communications software version. If, for example, this version is 3.6, the "highest revision" byte will be 3 and the "lowest revision" will be 6.

Transmission: [Dir][00][01][04][Chk]  
 Response: [01][03][Dir][00][Data 1][Data 2][Data 3][Chk]  
 Where: Data 1 = Communication protocol level  
 Data 2 = Part before the decimal point of the communications software version  
 Data 3 = Part after the decimal point of the communications software version  
 Dir = Address of the corresponding payout unit

Example:

The machine requests from payout unit number 1 information on the implementation level of the cctalk® protocol and the software version of the communications protocol.

Transmission: [03][00][01][04][F8H]  
 Response: [01][03][03][00][01][00][00][F8H]  
 Note: It is deduced from this response that the payout unit has level 1 of the cctalk® protocol implemented and that the software version of the communications software is 0.0.

About the payout unit [11H]

This command requests general information from the payout unit about the same.

Transmission: [Dir][00][01][11H][Chk]  
 Response: [01][04][Dir][00][Data1][Data2][Data3][Data4][Data5][Chk]  
 Where: Data 1 = Kind of device (the code for Hopper "U" will be "01")  
 Data 2 = Part before the decimal point of the software version  
 Data 3 = Part after the decimal point of the software version  
 Data 4 = Part before the decimal point of the communications software version  
 Data 5 = Part after the decimal point of the communications software version  
 Dir = Address of the corresponding payout unit

Example:

The machine requests from payout unit number 2 information about the kind of device in question, the software version and the communication protocol version.

Transmission: [04][00][01][11H][EAH]  
 Response: [01][05][04][00][01][02][03][01][04][FOH]  
 Note: It is deduced from this response that payout unit number 2 is a "Hopper U with cctalk® Communication" (code 01), with program version 2.3 and communication protocol version 1.4.

Status request [12H]

This command requests information from the payout unit about its current status.

Table of possible statuses

Code	Status	Data Bytes in Payout Unit Machine Frames
01H	Stand-by	1 byte
02H	Error	2 Bytes: error code
14H	Empty	3 bytes: Number of coins taken out
15H	Payout	5 bytes: 2 bytes for the number of coins paid out and another 2 for the number of coins pending to pay

If the payout unit is in "stand-by: "

Transmission: [Dir][00][01][12H][Chk]  
 Response: [01][01][Dir][00][Data 1][Chk]  
 Where: Data 1 = 01H = Payout unit in STAND-BY

If the payout unit is "emptying: "

Transmission: [Dir][00][01][12H][Chk]  
 Response: [01][03][Dir][00][Data 1][Data 2][Data 3][Chk]  
 Where: Data 1 = 14H = Payout unit emptying

Data 2 = High part of the number of removed coins counted by the meter

Data 3 = Low part of the number of removed coins counted by the meter

Dir = Address of the corresponding payout unit

If the payout unit is "paying out: "

Transmission: [Dir][00][01][12H][Chk]  
 Response: [01][05][Dir][00][Data 1][Data 2][...][Data 5][Chk]  
 Where: Data 1 = 15H = Payout unit paying out  
 Data 2/3 = High/low part of the number of paid-out coins counted by the meter  
 Data 4/5 = High/low part of the number of the pending coins to be paid  
 Dir = Address of the corresponding payout unit

Example:

The machine requests from payout unit number 2 information on its current status.

Transmission: [04][00][01][12H][E9H]

If the payout unit is in "stand-by",

Response: [01][01][04][00][01][FDH]

If the payout unit is "emptying",

Response: [01][03][04][00][14H][04][0AH][D6H]

Note: This response indicates that payout unit number 2 is in the process of emptying and that up to now it has emptied 1034 coins (040AH).

If the payout unit is "paying out",

Response: [01][03][04][00][15H][01][07][00][75H][66H]

Note: This response indicates that payout unit number 2 is in the process of paying out and that up to now it has paid 263 coins (0107H), with 177 coins left to pay (0075H).

Cancellation of the command being executed [13H]

This command causes the cancellation of the command that is being executed at that time. If the payout unit is in stand-by, it responds with a negative acknowledgement frame.

If the payout unit is in "stand-by", Transmission:

[Dir][00][01][13H][Chk]

Response: [01][00][Dir][05][Chk]

Where: Dir = Address of the corresponding payout unit.

If the payout unit is "emptying", transmission:

[Dir][00][01][13H][Chk]

Response: [01][03][Dir][00][Data 1][Data 2][Data 3][Chk]

Where: Data 1 = 14H = emptying payout unit  
 Data 2 = High part of the number of paid coins (8 bits)  
 Data 3 = Low part of the number of paid coins (8 bits)

Dir = Address of the corresponding payout unit

If the payout unit is "paying out", transmission: [Dir][00][01][13H][Chk]  
 Response: [01][05][Dir][00][Data1][Data2][...] [Data 5] [Chk]  
 Where: Data 1 = 15H = paying payout unit  
 Data 2 = High part of the number of paid coins (8 bits)  
 Data 3 = Low part of the number of paid coins (8 bits)  
 Data 4 = High part of the number of pending coins (8 bits)  
 Data 5 = Low part of the number of pending coins (8 bits)  
 Dir = Address of the corresponding payout unit

**Example:**

The machine requests from payout unit number 3 the cancellation of the command that is being executed at that time.

If the payout unit is in "stand-by,"	Transmission: [05][00][01][13][E7H] Response: [01][00][05][05][F5H]
If the payout unit is "emptying,"	Response: [01][03][05][00][14H][03][25H][BBH] Note: The payout unit has cancelled the emptying command, and at the time of cancellation, 850 coins had been extracted (0325H).
If the payout unit is "paying out,"	Response: [01][05][05][00][15H][00][41H][002][FOH][ADH] Note: The payout unit has cancelled the payout command, and at the time of cancellation, 65 coins had been extracted (0041H) and 752 remain to be extracted (02FOH).
If the payout unit were "emptying" or "paying out," and subsequent to the cancellation command a "status request" was sent, the payout unit would respond by communicating that the payout unit is in the stand-by status:	Transmission: [05][00][01][12H][E8H] Response: [01][01][05][00][01][F8H]

Empty [14H]

This command causes the payout unit to empty completely.

If the payout unit can execute the command, it returns a positive acknowledgement frame and begins to extract the coins until it detects a maximum span, it receives the cancellation command, it detects an error or a hardware reset is performed (power supply failure).

In the event that the payout unit cannot execute the command, it returns a negative acknowledgement frame and continues in the status it was in. The transmission of this negative acknowledgement may be due to the fact that there is an error on the payout unit or another command is being executed at that time.

Transmission: [Dir][00][01][14H][Chk]  
 Response: [01][00][Dir][00][Chk] Positive acknowledgement  
 Response: [01][00][Dir][05][Chk] Negative acknowledgement  
 Where: Dir = Address of the corresponding payout unit

**Example:**

The machine requests that payout unit number 2 empty all coins existing in its interior.

Transmission: [04][00][01][14H][E7H]  
 Response: [01][00][04][00][FBH]  
 Note: This response indicates that the payout unit has started executing the empty command.

**Pay out [15H]**

This command causes the pay out of the number of coins indicated in the bytes "Data 1" and "Data 2." These two bytes form a whole (number), where "Data 1" is the high part of the same and "Data 2" is the low part.

If the payout unit can execute the command, it returns a positive acknowledgement frame and begins to pay out coins until the indicated amount is extracted, until a maximum span is detected, until a cancellation command is received, until an error is detected or until the hardware is reset (power supply failure).

In the event that the payout unit cannot execute the command, it returns a negative acknowledgement frame and continues in the status it was in. The transmission of this negative acknowledgement may be due to the fact that there is an error on the payout unit or another command is being executed at that time.

Transmission: [Dir][02][01][15H][Data1][Data 2][Chk]  
 Response: [01][00][Dir][00][Chk] Positive acknowledgement  
 Response: [01][00][Dir][05][Chk] Negative acknowledgement  
 Where: Dir = Address of the corresponding payout unit

**Example:**

The machine requests that payout unit number 2 pay out 131 coins (0083H).

Transmission: [04][02][01][15H][00][83H][61H]  
 Response: [01][00][04][00][FBH]  
 Note: This response indicates that the payout unit has started executing the pay out command.

## Last status command [10H]

This command requests information from the payout unit about the "last command" executed and about the parameters of this command.

Last command table

Code	Status	Data bytes in Payout Unit Machine frames
01H	Stand-by	1 byte
14H	emptying	3 bytes: number of coins removed <sup>(3)</sup>
15H	Payout	5 bytes: 2 bytes for the number of coins paid out and another two for the number of coins pending to pay <sup>(4)</sup>

The first of the data bytes sent corresponds to the status itself of the payout unit.

(3) Data 1 is the high part of the whole (number) and Data 2 is the low part.

(4) Data 1 and Data 3 are the high part and Data 2 and Data 4 are the low part of their respective whole numbers.

There are three possibilities considered for "last command," which are the following: Stand-by, Emptying and Paying out. Until the software pertaining to EPROM handling is implemented, every time that a payout unit is powered up, the "last command" information will be stand-by.

Transmission: [Dir][00][01][10H][Chk]  
 Response: [01][01][Dir][00][Data 1][Chk]  
 Where: Data 1 = 01 = Payout unit in STAND-BY.  
 Dir = Address of the corresponding payout unit.

As from this moment, the "last command" information will be the last "pay out" or "empty" command.

If the "last command" executed was to "empty," the number of coins extracted (2 bytes) is also given.

Transmission: [Dir][00][01][10H][Chk]  
 Response: [01][03][Dir][00][Data 1][Data 2][Data 3][Chk]  
 Where: Data 1 = 14H = Last command executed was "empty."  
 Data 2/3 = High/low part of the number of removed coins counted by the meter  
 Dir = Address of the corresponding payout unit.

If the last command executed was to pay out, the number of coins paid out (2 bytes) and the number of coins pending payment (2 bytes) are also given.

Transmission: [Dir][00][01][10H][Chk]  
 Response: [01][05][Dir][00][Data 1][Data 2][...][Data 5][Chk]  
 Where: Data 1 = 15H = Last command executed was "pay out."  
 Data 2/3 = High/low part of the number of paid out coins counted by the meter.  
 Data 4/5 = High/low part of the number of coins to be paid.  
 Dir = Address of the corresponding payout unit.

This status request allows knowing the status of the payout unit after it finished executing the last command, but not how this command ended. This means that if the last command executed was a command to pay out 10 coins, we can find out that the command ended having paid out 6 of the 10 coins, but it does not give information about whether or not it ended due to the fact that it reached a maximum span or if a command to cancel the command was sent.

**Example:**

The machine requests information from payout unit number 1 on the last command executed.

	Transmission: [03][00][01][10H][Chk]
If the payout unit was in "stand-by,"	Response: [01][01][01][00][01][FCH] Note: This response indicates that the payout unit has not executed any command since it was put into operation.
If the payout unit was "emptying,"	Response: [01][03][01][00][14H][01][4CH][9AH] Note: This response indicates that the last command executed was to empty and that 332 coins were extracted.
If the payout unit was "paying out,"	Response: [01][05][01][00][15H][00][33H][00][2DH][84H] Note: This response indicates that the last command executed was to pay out and that 51 coins were paid, with 45 more remaining to be paid.

**Reset device [01H]**

This command causes the payout unit to reset the software. At that moment, the program execution of the payout unit goes to the reset vector.

The affected payout unit does not send any response to the machine.

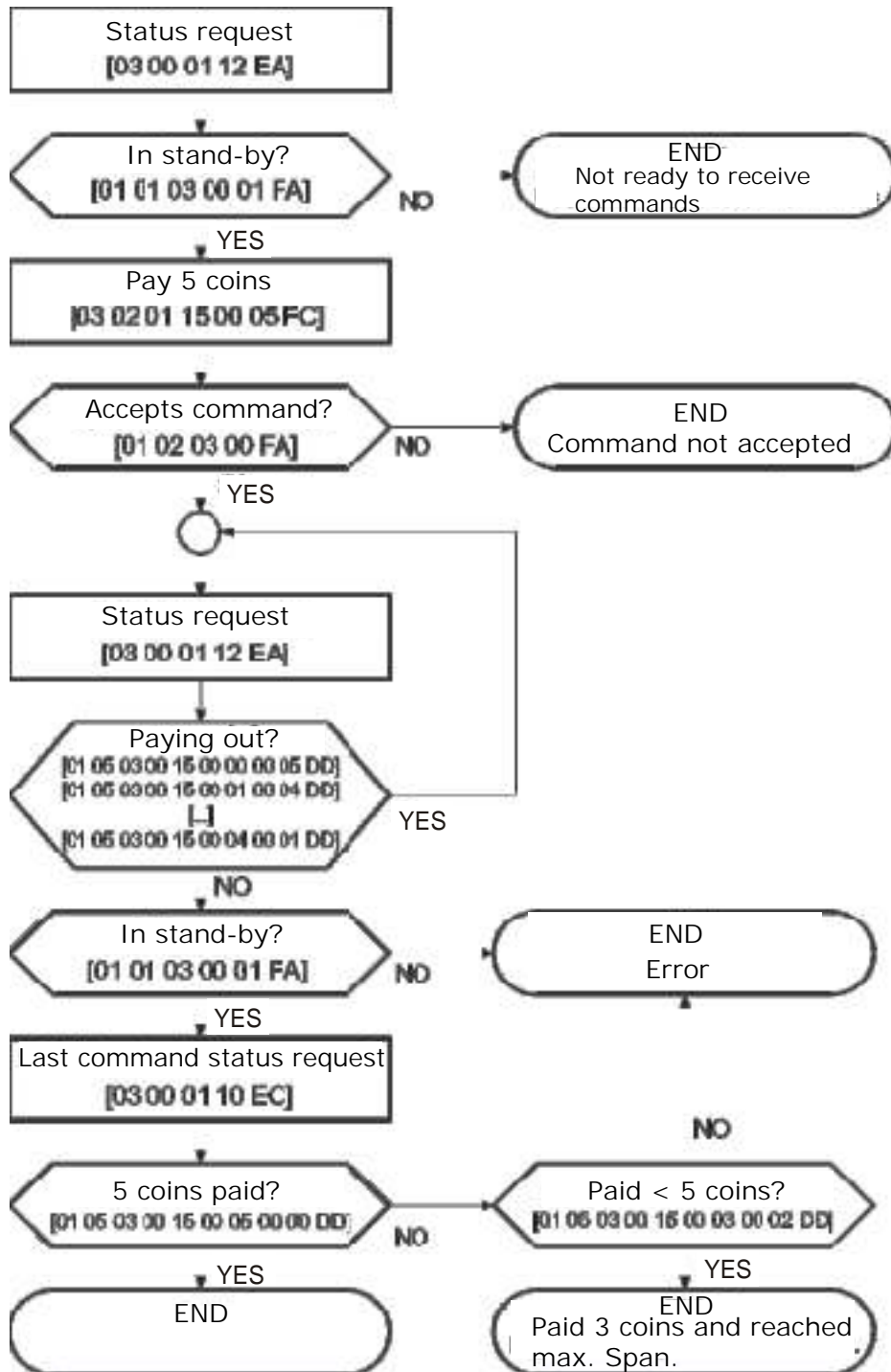
**Example:**

The machine requests that payout unit number 1 reset the software.

	Transmission: [03][00][01][01H][FBH]
Response:	None. The payout unit does not respond to this command.

1.3. Recommended programming algorithms

Algorithms for the payment of 5 coins by HOPPER 1.



Emptying algorithm of HOPPER 2.

