

Hopper CD + SV + iS3 coin counting system



User Manual



Via Cà Bianca, 421 - 40024
Castel San Pietro Terme (BO) - Italy

Progettazione e produzione di sistemi di pagamento e accessori per macchine Gaming, Vending e Car-Wash
Design and manufacture of payment systems and accessories for the Industries of Gaming, Vending and CarWash

Tel.: +39.051.944300
Fax.: +39.051.944594

Web: www.alberici.net
E.mail: info@alberici.net

1. General

Congratulations on the purchase of the U.C.S. counting system for massive mixed coins, developed and produced by Alberici SpA. This system is designed in such a way as to quickly check and count large amounts of coins.

It is equipped with iS2 or iS3 sorter, to separate the counted coins into 2 or 3 different directions (dispensing hoppers or coin boxes).

Rejected coins are ousted directly by the coin acceptor.

It get easily installed in Cross-Change machines, where it is necessary to count and recycle large quantities of.



1.1 Operation

The U.C.S. massive coin counting unit can identify various currencies and coin denominations, poured in as mixed heaps, provided that the diameter of such coin ranges between $18 \div 26.5$ mm, and that their thickness stands between $1.8 \div 2.6$ mm.

The system consists of the following components, all of them working by the ccTalk protocol:

- U.C.S. feeder hopper:** receives the heap of coins to be counted, and starts one-by-one delivery to the inlet of the SV fast coin acceptor.
- SV (super high speed) cctalk coin validator,** capable of processing and identifying up to 10 coins per second.
- 2-way iS2 or 3-way iS3 coin sorter (optional):** it communicates with the system via its Pulse 10p connector.

1.2 Safety

The U.C.S. system must be installed in machines equipped with devices apt to CUT power supply off.

Always power off before removing or installing the device .

The system contains moving mechanical parts: **DO NOT INTRODUCE** fingers or objects other than coins while the device is operating, or when power is switched on.



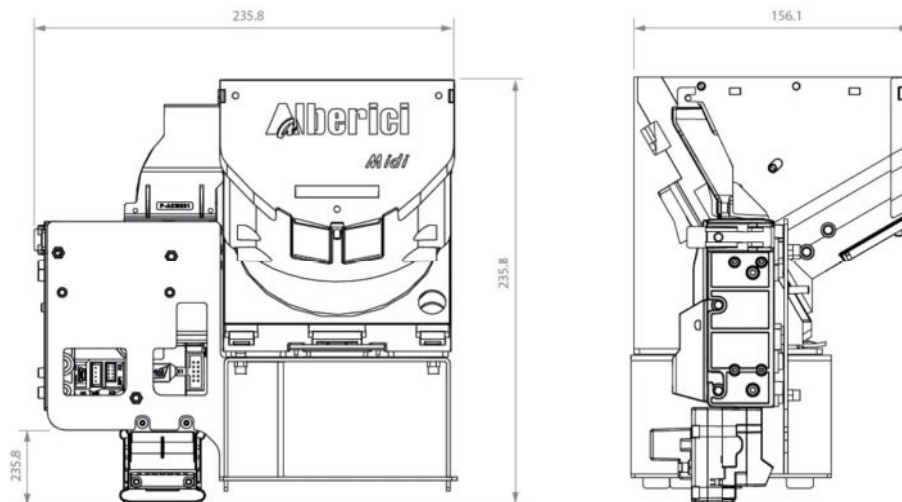
**PAY ATTENTION:
DANGER!**
MECHANICAL PARTS IN MOTION

2. Technical specs

Weight	2.250 kg (with iS2), 2.140 kg (without iS2)
Voltage	24 Vdc
Current draw	1.70 A
Interfaces	CcTalk, SPI
Operating temperature	0°C ÷ 75°C
Humidity	20% ÷ 75%
Speed	280 coins/min (7-Holes); 240 coins/min (6-Holes); 200 coins/min (5-Holes)
Coin capacity	500 mixed coins
Coins diameter	$18 \div 26.5$ mm
Coins thickness	$1.8 \div 2.6$ mm

3. Mechanical description

3.1 Dimensions



Counting Unit	A-CM0073
Counting Unit + iS2 coin sorter	A-UC0002
Counting Unit + iS3 coin sorter	A-CM0087

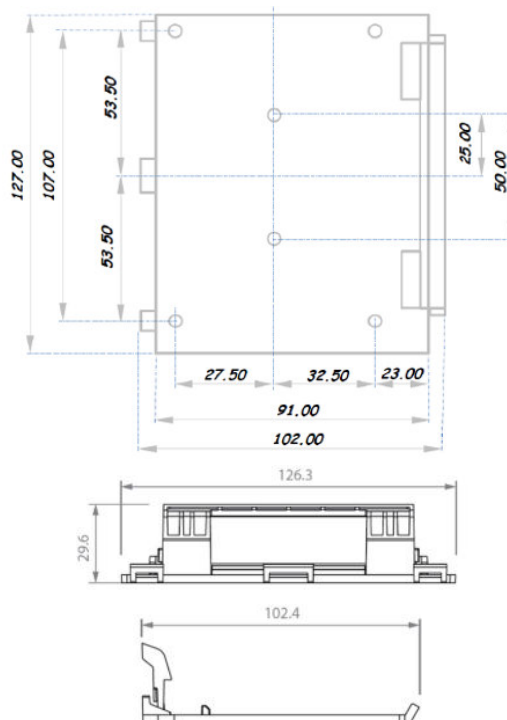
3.2 Installation

Fasten the polycarbonate base onto the support prepared in the cabinet for the hopper of the UCS unit.

Position the hopper on the base, keeping the reservoir towards the front the release tabs of the base, and push it down.

Before connecting the power supply to the 10p socket located at the rear side of the hopper, please read chapter 4.

To remove the hopper, disconnect the connection cable, pull the release tab out and pull the hopper upwards.



For the guarantee to be valid Installation must be done in compliance with the instructions given in the present section 3.2.

4. Electrical description

4.1 Power supply

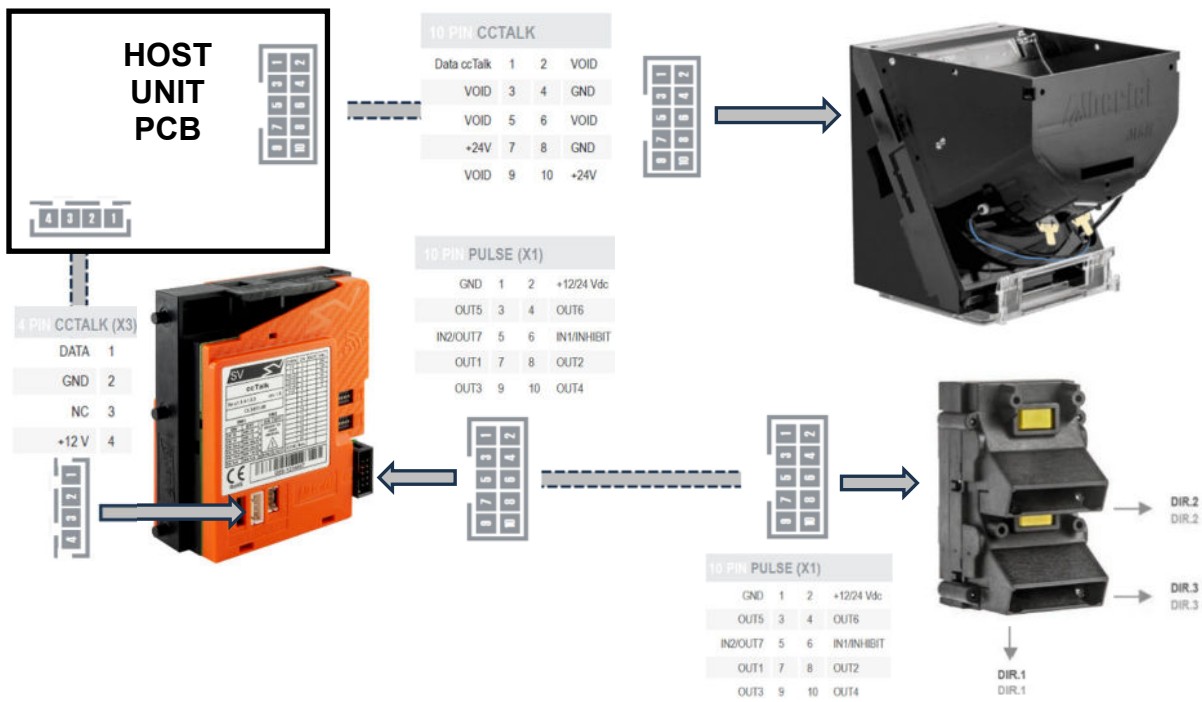
Power supply to Hopper U.C.S. must be 24V d.c. . Leads cross section must be compatible with current draw showed in section 2.

4.2 Pin-outs

The Hopper cctalk 10p connector is located in the rear side of the hopper slide base, beside the dip-switch row for setting the serial address.

The cctalk communication between the machine host and the U.C.S. system runs through this 10p connector.

Connect to Ground (GND) all the plate terminals of the hopper.



4.3 Setting the serial address by Dip-Switches

If necessary, the serial address of the hopper can be modified via the Dip-Switch row. The three dip-switches can be combined as follows, to set the needed address:


Default address is 3 (all switches to OFF)			
SW1 position	SW2 position	SW3 position	Serial Address
OFF	OFF	OFF	3
ON	OFF	OFF	4
OFF	ON	OFF	5
ON	ON	OFF	6
OFF	OFF	ON	7
ON	OFF	ON	8
OFF	ON	ON	9
ON	ON	ON	10

The status of the dip-switches gets detected only at power up or at reset, therefore any setting will not have any effect until after the unit is switched off and on again.

5. Maintenance

Before any maintenance operation that requires disassembling the system, turn off the power and disconnect the cable.

Watching from the coin insertion basket, check if debris or rubbish are laying at the bottom of the dispenser belt, and remove them (if necessary, use a jet of compressed air). Their presence can obstruct the output of the coins, hinder chain movement, spoil the components of the U.C.S. or alter its performance.

It is good practice to periodically activate the automatic cleaning button : the hoper will start the belt in the direction opposite to the standard rotation, thus forcing it to be cleaned by the built-in brush. Keep the button pressed for as long as necessary. It is also useful to blow compressed air onto the slots for draining debris and liquids (see the picture on page 4), to release any obstructions. Carry out the same operation on the holes of the channel between the hopper outlet and the coin validator input.

The coins will get into the coin acceptor already abundantly treated clean; however, over time it is possible to accumulate debris in the acceptor reading path. The reading path can be accessed by opening the side flap door of the acceptor: remove the debris that have been deposited by blowing compressed air, and then rub the walls with a soft clean cloth, lightly soaked in a cleaning solution for glass and plastic surfaces.

CAUTION:

- 1) do not use alcohol, diluent, turpentine essence, acetone, petrol or other petroleum products, or containing citric acid.
- 2) the lenses of the sensors are made of transparent polymers, and must be cleaned with great caution so as not to scratch them. Gently wipe several times, until the dirt is removed.

6. ccTalk protocol

ccTalk® communication protocol is the Money Controls (formally Coin Controls) serial communication protocol for low-speed control networks. It was designed to allow the interconnection of various cash handling devices (*Hopper, Card reader, Bill validators, Coin selectors etc.*), mostly in AWP and gaming Industry, but also in other devices that use those components.

ccTalk® is an open standard. All documentation is available at web site: www.cctalk.org.

Communication protocol of Alberici U.C.S. is implemented according to generic specification 4.4

1 Communication specifications

ccTalk serial communication is derivation of RS232 standard.

Low data rate NRZ (*Non Return to Zero*) asynchronous communication:

Baud rate 9600, 1 start bit, 8 data bits, no parity, 1 stop bit.

RS232 handshaking signals (*RTS, CTS, DTR, DCD, DSR*) are not supported. Message integrity is controlled by means of checksum calculation.

1.1 Baud rate

The baud rate of 9600 was chosen as compromise between cost and speed. Timing tolerances is same as in RS232 protocol and it should be less than 4%.

1.2 Voltage level

To reduce the costs of connections the "Level shifted" version of RS232 is used. The idle state on serial connector is 5V, and active state is 0V.

Mark state (*idle*) +5V nominal from 3.5V to 5V

Space state (*active*) 0V nominal from 0.0V to 1.0V

1.3 Connection

The connection of Hopper at network is achieved by means of 10 pole IDC connector compatible with standard ccTalk 2+2 (Two wires for power supply + two for GND connector). Connector is used for power supply and communication as well.

For schematics and connector lay-out see image and table below.

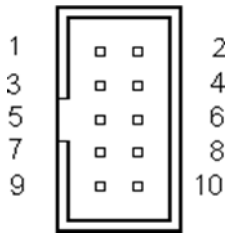


Fig. 1 - ccTalk connector lay-out

PIN Nr.	Function
1	ccTalk Data
2	Not used
3	Not used
4	GND
5	Not used
6	Not used
7	+24 V
8	GND
9	Not used
10	+24 V

10p ccTalk
connector
pin-out

1.4 Message structure

Each communication sequence consists of two message packets.
Message packets for simple checksum case is structured as follows:

[Destination address]
 [Nr. of data bytes]
 [Source address]
 [Header]
 [Data 1]
 ...
 [Data n]
 [Checksum]

There is an exception of message structure when device answer to instruction 253 "Address poll" and 252 "Address clash". The answer consists of only one byte representing address delayed for time proportional to address value or random delay.

For CRC checksum case format is:

[Destination address][Nr. of data bytes]
 [CRC 16 LSB]
 [Header][Data 1]
 ...
 [Data n]
 [CRC 16 MSB]

1.4.1 Address

Address range is from address 0 to address 255. Address 0 is special case or so called "broadcast" address and address 1 is default host address.

The recommendations for address value of different devices are presented in table below.

Device category	Address	Additional addr.	Note
Coin Acceptor	2	11 - 17	Coin validator, selector, mech...
Payout	3	4 - 10	Hopper
Bill validator	40	41 - 47	Banknote reader
Card Reader	50		-
Display	60		Alphanumeric LC display
Keypad	70		-
Dongle	80	85	Safety equipment
Meter	90		Replacement for el.mec. counters
Power	100		Power supply

1.4.2 Number of data byte

Number of data byte in each transfer could be from 0 to 252. Value 0 means that there are no data bytes in the message, and total length of message packet will be 5 bytes. Although theoretically it will be possible to send 255 bytes of data because of some limitations in small micro controllers the number is limited to 252.

1.4.3 Command headers (*Instructions*)

Total amount of possible ccTalk command header is 255, with possibility to add sub-headers using headers 100, 101, 102 and 103.

Header 0 stands for **ACK** (*acknowledge*) replay of device to host. **Header 5** stands for **NAK** (*No acknowledge*) replay of device to host. **Header 6** is **BUSY** replay of device to host.

In all three cases no data bytes are transferred. Use of ACK and NAK headers is explained separately for each specific message transfer.

Commands are divided in to several groups according to application specifics:

- Basic general commands
- Additional general commands
- Commands for Coin acceptors
- Commands for Bill validators
- Commands for Payout
- MDCES commands

1.4.4 Data

There is no restrictions for data format use. Data could be BCD (*Binary Coded Decimal*) numbers, Hex numbers or ASCII strings. Interpretation as well as format is specific to each header use, and will be explained in separate chapter.

1.4.5 Checksum

Message integrity during transfer is checked by use of simple zero checksum calculation. Simple checksum is made by 8 bit addition (modulus 256) of all the bytes in the message.

If message is received and the addition of all bytes are non-zero then an error has occurred..

1.5 Timing specification

The timing requirements of ccTalk are not very critical but there are some recommendations.

1.5.1 Time between two bytes

When receiving bytes within a message packet, the communication software must wait up to **50 ms** for next byte if it is expected. If time out occurs, the software should reset all communication variables and get ready to receive next message. The inter byte delay during transmission should be ideally **less than 2 ms** and **not greater than 10 ms**.

1.5.2 Time between command and replay

The time between command and reply is dependent on application specific for each command. Some commands return data immediately, and maximum time delay should be within **10 ms**.

Other commands that must activate actions in device may return reply after action is finished.

1.5.3 Start-up time

After the power-up sequence Hopper should be ready to accept and answer to a ccTalk message within less than 250 ms. During such period all internal check-up and system settings must be done, and hopper should

1.6 Error handling

If slave device receive the message with bad checksum or missing data no further action is taken and receive buffer will be cleared. Host software should decide to re-transmit message immediately or after a fixed amount of time. In case when host receive message with error it has same options.

2. Hopper CD Command header set

Command header set, that host could use in communication with Hopper is given in table below.

Code / Hex.	Command header	Note
254	FE	Simple poll
253	FD	Address poll
252	FC	Address clash
246	F6	Request manufacturer id
245	F5	Request equipment category id
244	F4	Request product code
242	F2	Request serial number
241	F1	Request software revision
217	D9	Request payout high/low status
210	D2	Modify sorter path
197	C5	Calculate ROM checksum
164	A4	Enable hopper
163	A3	Test hopper
134	86	Dispense hopper value
133	85	Request hopper polling value
132	84	Emergency stop value
131	83	Request hopper coin value
130	82	Request indexed hopper dispense count
1	1	Reset device

Command headers are divided into 4 different groups:

- Common command headers
- Hopper command headers
- MDCES command headers

U.C.S. SYSTEM SPECIFIC COMMAND HEADERS

The components of the U.C.S. System interact as follows.

- 1) The machine host prompts the hopper for dispensing the coins it contains. The hopper starts paying coins out.
- 2) The machine host asks the coin acceptor to inform the denomination processed.
- 3) If the coin acceptor communicates any error, the cctalk automatic reject motor opens and closes the acceptor side flap.

Specific commands for the Hopper CD:

242 - Request serial number

164 - Enable hopper

167 - Dispense hopper coin

166 - Polling – request hopper status (aim being to understand if belt is rotating or not)

163 - Test hopper

172 - Stop

Specific commands for the SV coin acceptor:

231 - Enable coins (modify inhibit status)

210 - (Optional) Modify sorter path

229 - Polling (to identify the accepted denomination or whether the acceptor is in error)

2.1 Commands for Hopper CD:

Default address = 03

2.1.1 Command header 242 [hexF2], Request serial number

Hopper answer with three-byte serial number.

Message format is:

Host sends: [Dir] [00] [01] [F2] [Chk] Hopper answer: [01] [03] [Dir] [00] [Serial 1 -LSB] [Serial 2] [Serial 3 -MSB] [Chk] Serial 1 – first data byte sent is LSB of serial number.

Example of message packets for Hopper (address 3) and serial number 1-2-34567, hex [BC][61][4E] is:

Host sends: [02] [00] [01] [F2] [0A] Hopper answer: [01] [03] [03] [00] [4E][61][BC] [8E]

2.1.2 Command header 164 [hexA4], Enable Hopper

This command enable hopper before paying out coin.

Command format is:

Host sends: [Dir][01][01] [A4] [d1][Chk]

Hopper answer: [01] [00] [Dir] [00] [Chk] ACK

d1 must be Hex [A5] in order to enable hopper.

Example of message packets for Hopper (address 3) is

Host sends: [03][01][01] [A4] [A5][B2]

Hopper answer: [01] [00] [03] [00] [FC] ACK

2.1.3 Command header 167 [hexA7], Dispense hopper coin

This command dispenses coin from the hopper. Maximum number of coin hopper can dispense with a single command is 255. After Dispense hopper coin command, hopper need to be enabled, else dispense action is not performed. Alberici hopper answers correctly to two formats of dispense coin command.

First message format is

Host sends: [Dir] [04] [01] [A7] [sn1] [sn2] [sn3] [N°Coin][Ch k] Hopper answer: [01] [00] [Dir] [00] [Chk] ACK or NAK

Example of first type of message packets for Hopper (address 3) is

Host sends: [03] [04] [01] [A7] [12] [34] [56] [64][Chk] Hopper answer: [01] [00] [03] [05] [F7] NAK

Command tries to pay 100 coins (64H) but serial number sent to hopper isn't correct.

Second command format is

Host sends: [Dir][0A][01] [A7] [00] [00] [00] [00] [00] [00] [00] [00] [00] [N°Coin][Chk] Hopper answer: [01] [00] [Dir] [00] [Chk] ACK or NAK

Example of second type of message packets for Hopper (address 3) is

Host sends: [03][09][01] [A7] [00] [00] [00] [00] [00] [00] [00] [00] [01][4B]

Hopper answers: [01] [00] [03] [00] [FC] ACK

One token is paid

2.1.4a Command header 166 [hexA6], Request hopper status

This command returns four counters that explain the status of payment.

Host sends: [03][09][01] [A7] [00] [00] [00] [00] [00] [00] [00] [00] [01][4B]

Hopper answer: [01] [00] [03] [00] [FC] ACK

One token is paid.

2.1.4 b Command header 166 [hexA6], Request hopper status

polling only to understand if it is moving – does not check the paid-out coins)

This command returns four counters that explain the status of payment.

Host sends: [03][09][01] [A7] [00] [00] [00] [00] [00] [00] [00] [00] [01][4B]

Hopper answer: [01] [00] [03] [00] [FC] ACK

One token is paid.

These four bytes are:

1. Event Counter that show the number of good dispense events since last reset.
2. Payout coins remaining that show how many coins are still to pay.
3. Last Payout: coins paid, shows how many coins paid out since last dispensing command (increments with each coin dispensed)
4. Last Payout: coins unpaid, that show how many coins remained unpaid during last payout.

First two counters are saved in Ram, while last two are saved in eeprom. Default value of Event Counter and Payout coins remaining is 0, at reset and after Emergency stop command. If a reset occurs, Event Counter and Payout coins remaining values are saved in two Last Payout counters, in eeprom. Thus, after reset or power-off, hopper can return coin paid and unpaid during last payout. Command format is

Host sends: **[Dir] [00] [01] [A6] [Chk]**

Hopper answer: **[01] [04] [Dir] [00] [d1] [d2] [d3] [d4] [Chk]**

Example of message packets for Hopper (address 3) is

Host sends: **[03] [00] [01] [A6] [56]**

Hopper answer: **[01] [04] [03] [00] [00] [00] [07] [03] [EE]**

In this example hopper is not perform a payout. During last payout the hopper was power off while paying. It had to pay 10 coin, but only 7 was really paid. Three remained.

Another example of message packets for Hopper (address 3) is

Host sends: **[03] [00] [01] [A6] [56]**

Hopper answer: **[01] [04] [03] [00] [0B] [09] [02] [00] [E2]**

In this example hopper is performing a payout. It's the 11th payout before last reset. A coin is paid (9 are remaining) and during last payout 2 coin was paid.

2.1.5 Command header 163 [hexA3], Test Hopper

This command is used to test hopper hardware. It reports back a bit mask that shows various hopper errors. Bit meaning is shown here :

BIT0 – Absolute maximum current exceeded BIT1 –

Payout timeout occurred BIT2 – Motor reverse during last payout to clear a jam BIT3 – Opto fraud attempt, path blocked during idle BIT4 – Opto fraud attempt, short circuit during idle BIT5 – Opto blocked permanently during payout BIT6 – Power up detected

BIT7 – Payout disabled Command format is

Host sends: **[Dir][00][01] [A3][Chk]**

Hopper answer: **[01] [00] [Dir] [00] [d1] [d2] [Chk]** Example of message packets for Hopper (address 3) is Host sends: **[03][00][01] [A3][59]**

Hopper answer: **[01] [02] [03] [00] [C0] [00] [3A]**

The data byte Hex[60] means that Opto are blocked permanently during payout and power up was detected.

2.1.6 Command header 172 [hexAC], Emergency stop.

This command immediately stops the payout sequence and reports back the number of coins which the hopper failed to pay out. After Emergency stop command hopper is disabled. To perform new payout sequence, hopper must be re-enabled.

Message format is:

Host sends: **[Dir] [00] [01] [AC] [Chk]**

Hopper answer: **[01] [01] [Dir] [00] [d1] [Chk]**

Example of message packets for Hopper (address 3) is

Host sends: **[03] [00] [01] [AC] [50]**

Hopper answer: **[01] [01] [03] [01] [01] [FA]**

Data byte Hex[01] mean that hopper remain one coin to be paid.

2.2 Commands for the coin acceptor

Default address = 02

2.2.1 Command 231 [hexE7], Modify inhibit status

With this command host can inhibit the acceptance of some or all coins.

Acceptance or inhibition is set with a two-byte mask sent by host.

Bits from 0 to 15 determinate coin positions from 1 to 1622.

Number of coin channels in new ALBERICI coin selectors AL55/66 is same as number of position (16). Message format is:

Host sends: **[Dir] [02] [01] [E7] [LSB Mask.] [MSB Mask.] [Chk]**

Coin s. respond: **[01] [00] [Dir] [00] [Chk] ACK**

Example of message string to enable all position for coin selector(address 2) is:

Host sends: **[02] [02] [01] [E7] [FF] [FF] [16]**

Coin s. respond: **[01] [00] [02] [00] [FD] ACK**

After that all programmed coins will be enabled. Command has no effect on coin position that are not programmed. Initially coin channels could be programmed with acceptance enabled or disabled.

2.2.2 Command 210 [hexD2], Modify sorter paths (optional)

With this command the host can change direction of coins in sorter if sorter is supported. Host sends two bytes of data to select the coin position and sorter path (direction of exit). First byte of data (LSB) represents coin position and second byte of data points to sorter path. ALBERICI coin selectors has support for most existing sorters that have direct drive of coils from coin selector with open collector transistor. Most common

are 3- or 4-way sorter with two coils, but recently 5-way sorters with 3 coils are in use. Message format is: Host sends: **[Dir] [02] [01] [D2] [Coin pos.] [Sort.Path] [Chk]**

Coin s. respond: **[01] [00] [Dir] [00] [Chk]** ACK if sorter path is possible to set

Coin s. respond: **[01] [00] [Dir] [05] [Chk]** NAK if coin selector does not support setting

Initially all coin position has sorter paths set to direction 1 hex[01]. If sorter is not supported, sorter path is set initially to 0 hex[00]!

If host sends command to modify sorter path that is not existent or for coin not programmed, the coin selector will respond with message NAK. Example of message string for coin selector (address 2) re-direction of coin **pos. 1** into **path 2** is:

Host sends: **[02] [02] [01] [D2] [01] [02] [26]**

Coin s. respond: **[01] [00] [02] [00] [FD]** ACK

After acceptance of command, accepted coins with position 1 will exit in direction 2 of the sorter. The path or direction 1 is usually one without activation of any coil.

Different coil activation schematics is possible to program by setting the sorter type.

2.2.3 Command 229 [hexE5], Read buffered credit or error codes

This is the most important command used by host to detect import of coins in to a machine and to report eventual errors. As previously mentioned, coin selectors store all events in volatile memory called credit buffer. Buffer has 5 level and use two bytes for each event. In first byte coin position or coin value¹ is stored.

The second byte point to a sorter path or indicate error code.

If during coin acceptance any error occurs, stored value of coin position is 0, hex [00].

Error codes supported in ALBERICI coin selectors are shown in Table 4 next page.

Code d.	Code h.	Error	Coin rejected
0	00	Null event	No
1	01	Reject coin (not recognized)	Yes
2	02	Inhibited coin (master inhibit)	Yes
3	03	Multiple window (fraud or similar coin)	Yes
5	05	Validation (measuring) time out	Yes
6	06	Credit sensor (recognition to opto 2) time out	Possible
8	8	Second close coin	Yes/both
16	10	Credit sequence error (Yo-yo)	No
18	12	Coin too fast (opto 2 minimum time not elapsed)	No
19	13	Coin too slow (opto 2 time out)	No
128	80	Inhibited coin (position 1)	Yes
...	...	Inhibited coin (position n)	Yes
143	8F	Inhibited coin (position 16)	Yes
255	FF	Unspecified alarm code	-

¹ If coin selector use CVF (Coin Value Form)

Table 4 Acceptance error codes

Coin selectors also use one eight-bit counter² that is incremented each time a new coin is detected. At the same time data in coin credit buffer are shifted two positions to the right. When counter reaches the value of 255 it toggle to a value 1 and continue to increment on each event. Event counter is set to value "0" after each power-up or acceptance of reset command. The first two byte (LSB) in coin credit buffer always contain the data of last event. Host software must read event counter and coin credit buffer data in period short enough to prevent the loss of coin data³. Message format is:

Host sends: [Dir] [00] [00] [E5] [Chk]

Coin sorter responds: [01][0B] [Dir] [00] [Ev.cnt.][coin code 1][dir/err] [coin code 2][dir/err] . . . [coin code 5][dir/err] [Chk]

Examples of message string for coin selector (address 2) after coin insertions:

Host sends: [02] [00] [00] [E5] [18] Polling minimum each 500 ms

Coin sorter respond: [01] [0B] [02] [00] [00][00][00][00][00][00][00][00][00][00] [F2]

The respond after power-up or reset

Coin sorter respond: [01] [0B] [02] [00] [01][01][02][00][00][00][00][00][00][00][00] [EE]

First event, coin position 1, sorter path 2 accepted

Coin sorter respond: [01] [0B] [02] [00] [02][02][01][01][02][00][00][00][00][00] [EA]

Second event, coin position 2, sorter path 1 accepted

Coin sorter respond: [01] [0B] [02] [00] [03][00][02][02][01][01][02][00][00][00] [E7]

Third event, coin rejected due to master inhibit active

Coin sorter respond: [01] [0B] [02] [00] [04][00][83][00][02][02][01][01][02][00][00] [63]

Forth event, coin position 4 inhibited and rejected

From example we can notice shifting of data in the coin credit and error buffer and increment of event counter.

2.3 Commands referring to the generic Algorithm:

Read the the serial number of the U.C.S. (very important; otherwise it is possible to make it dispense)

[242] Enable the U.C.S. hopper (very important; otherwise it is possible to make it dispense)

[164] Enable the coins accepted by the coin selector (otherwise it will reject all coins)

Set the separator sorting paths for the accepted coins

[210] Start coin dispensing by U.C.S. hopper (request 250 coins payout) [134

Cyclically (at least every 500ms, ideally every 150ms), it is recommended to:

- check the dispensing status of the hopper [166]

- check the coins accepted by the coin acceptor [229]

It is possible to exit the counting cycle under the following conditions:

- when the requested number of coins has been dispensed, it may be necessary to command a further dispensing cycle, because the number of coins contained in the hopper could be larger than the maximum number we can ask for by each command (see command 166);

- the hopper has stopped without reaching the required number of coins. In this case it must be investigated whether the hopper operation has exceeded time-out (empty tank)limit or because of an error. To determine which cause, make use of the Test hopper command (see command 163);

- the coin validator has answered an error code (see command 229. In this case, send the hopper Stop command (command 172). Should any error occur, the coin reject mechanism can be activated (command FF-11).

IMPORTANT: the coins accepted by the system must be the same ones confirmed by the coin acceptor as per command 229.

3.0 Setting the Hopper Address via the DIP-SWITCH row in the 10p pcb

The default address of Alberici hoppers (03) can be changed by setting the onboard switches. The following table shows the possible Dip-Switch combinations that can be used to set the Hopper address.

Sw 3	Sw 2	Sw 1	Address
Off	Off	Off	3
Off	Off	On	4
Off	On	Off	5
Off	On	On	6
On	Off	Off	7
On	Off	On	8
On	On	Off	9
On	On	On	10

The board reads the address dip-switches only after power-up or reset. Therefore any change of address made by means of the dip-switch row during normal operation will have no effect.

DICHIARAZIONE DI CONFORMITÀ



DIRETTIVA 2014/35/CE - DIRETTIVA 2014/30/UE

La ditta Alberici S.p.A., avente sede in via Ca' Bianca 421, 40024 Castel San Pietro Terme (BO) – Italia,

DICHIARA

Che il sistema classificato nella famiglia di prodotto **apparecchio elettrico d'uso domestico e similare – dispositivo elettromeccanico di validazione e conteggio delle monete**, identificato in modo seguente da:

Modello Configurazione Numero di matricola
AVENTADOR **ccTALK 24V** -----

Essendo realizzato conformemente al modello denominato AVENTADOR, finito di testare con esito positivo ai fini EMC e LVD (rapporto 7416-AVENTADOR.doc), dalla STP S.r.l., con sede legale in via P.F. Andrelini, 42,47121 Forlì (FC), Italia, e sede operativa in via San Donnino, 4, 40127 Bologna (BO), Italia, risulta essere conforme a quanto previsto dalle seguenti norme comunitarie:

- a) le norme armonizzate (per i punti applicabili):
- CEI EN 55014-1 (CEI 110-1);
 - CEI EN 55014-2 (CEI 210-47);
 - CEI EN 55022 (CEI 110-5);
 - CEI EN 55024 (CEI 210-49);
 - CEI EN 60065 (CEI 92-1);
 - CEI EN 60335-1 (CEI 61-150);
 - CEI EN 60335-2-82 (CEI 61-210);
 - CEI EN 60950-1 (CEI 70-200);
 - CEI EN 61000-3-2 (CEI 110-28);
 - CEI EN 61000-3-3 (CEI 110-28);
 - CEI EN 61000-4-2 (CEI 210-34);
 - CEI EN 61000-4-3 (CEI 210-39);
 - CEI EN 61000-4-4 (CEI 210-35);
 - CEI EN 61000-4-5 (CEI 110-30);
 - CEI EN 61000-4-11 (CEI 110-29);
 - CEI EN 61000-6-1 (CEI 210-64);
 - CEI EN 62233 (CEI 61-251).
- b) In conformità ai requisiti di sicurezza della Direttiva Bassa Tensione:
- L. 791 del 1977 e s.m.
 - 2011/65/UE del 26 Febbraio 2014;
- c) in conformità ai requisiti essenziali di sicurezza della Direttiva Compatibilità Elettromagnetica:
- 2004/108/CE del 06 Novembre 2007
 - 2014/53/UE del 26 Febbraio 2014;

che conferisce presunzione di conformità alla Direttiva 2014/30/UE

Castel San Pietro Terme (BO), Italia li, ___/___/___

Felizio Alberici

Il Presidente



NOTA: La Alberici S.p.A. si riserva il diritto di apportare modifiche alle specifiche tecniche dell'apparecchiatura descritta in qualunque momento e senza preavviso, nell'ambito del perseguimento del miglioramento continuo del proprio prodotto.



Via Cà Bianca, 421 - 40024
Castel San Pietro Terme (BO) - Italy

Progettazione e produzione di sistemi di pagamento e accessori per macchine Gaming, Vending e Car-Wash
Design and manufacture of payment systems and accessories for the Industries of Gaming, Vending and CarWash

Tel.: +39.051.944300
Fax.: +39.051.944594

Web: www.alberici.net
E.mail: info@alberici.net