Sistema combinato AVENTADOR

Manuale d'uso

CE

Alberici

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

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

Tel.: +39.051.944300
Fax.: +39.051.944594

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

1. Introduzione

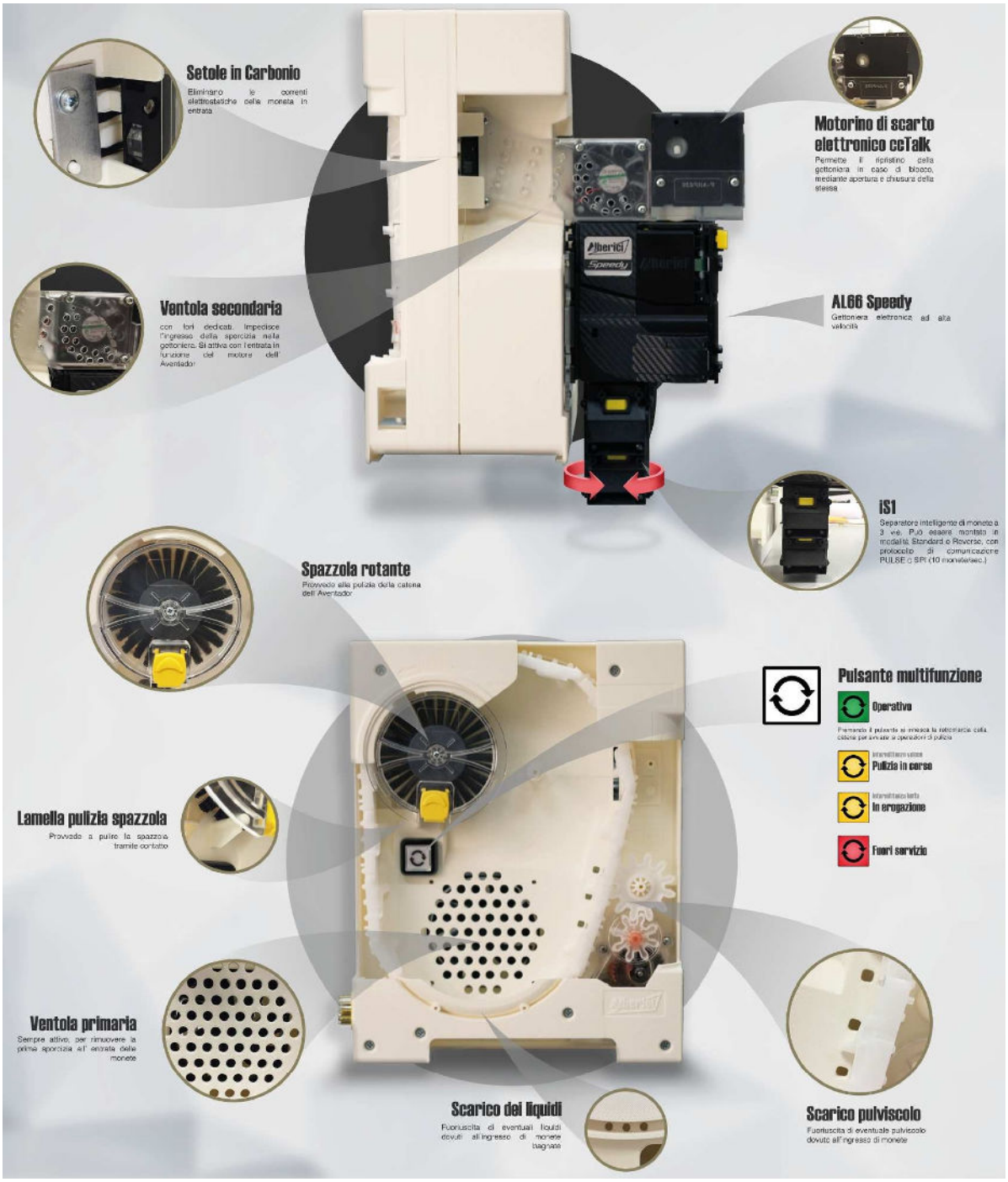
Congratulazioni per l'acquisto del Sistema AVENTADOR per il conteggio massivo di monete, sviluppato e prodotto da Alberici SpA.
Il sistema è stato progettato per soddisfare le più severe esigenze di precisione e sicurezza nella gestione dei pagamenti.
E' concepito in modo tale da eliminare gli interventi di manutenzione ordinaria. Ogni qualvolta entra in funzione per validare e contare le monete introdotte, il dispositivo esegue automaticamente varie operazioni di pulizia.
Può essere corredato con il separatore intelligente iS3, per dividere le monete contate in 3 direzioni diverse.
Le monete rifiutate vengono espulse separatamente.
Si installa facilmente in **macchine Cambia-Cambia**, ove è necessario contare e riciclare consistenti quantità di monete in una singola sessione.

1.1 Funzionamento

L'unità di conteggio massivo Aventador Alberici S.p.A. può riconoscere e contare vari tipi e valori misti di moneta, di diametro compreso fra 21mm e 31,5mm (a richiesta 16mm-24mm), e di spessore compreso fra 1,7 mm e 3,4mm.
Il sistema è composto dai seguenti componenti coordinati fra loro in protocollo ccTalk:

a. ***Erogatore feeder auto-pulente Aventador***: riceve la massa di monete da contare, e le fornisce una alla volta all'ingresso della gettoniera. L'erogatore è dotato di vari elementi (*) dedicati alla pulizia della catena e all'eliminazione di residui, pulviscolo e cariche elettrostatiche dalla superficie delle monete da contare.
b. ***Gettoniera Superveloce SV***, capace di processare e identificare fino a 10 monete al secondo.
c. ***Kit elettronico di ripristino automatico*** della gettoniera: in caso di problemi di inceppamento meccanico, il congegno attiva l'apertura dello sportello basculante della gettoniera, provocando l'eliminazione delle monete inceppate attraverso il canale di scarto, naturalmente senza contarle.
d. ***Separatore intelligente di monete iS1 a tre vie (opzionale)***: comunica con il sistema attraverso la porta SPI o attraverso il connettore 10 poli, riportando eventuali errori di indirizzamento o di funzionamento, in modo da poter arrestare tempestivamente il processo di conteggio.

(*) Cfr. illustrazione nella pag. seguente:
1. Il cestello di introduzione è dotato di fori per lo scolo dei liquidi e delle impurità grossolane.
2. Sul cestello soffia una ventola che separa gli oggetti volatili (es. mozziconi di sigaretta, brandelli di carta, lanugini e batuffoli di tessuto, ecc. ecc.) dalle monete.
3. I residui più piccoli e pesanti cadono verso le feritoie di espulsione presenti sul fondo, e lungo il binario della catena.
4. La catena striscia contro una spazzola che asporta i residui e riduce la presenza di grassi. A sua volta la spazzola rotante viene tenuta pulita da una lamina a contatto.
5. Le monete in uscita strisciano contro una spazzolina di setole in carbonio, che annullano le eventuali cariche elettrostatiche presenti.
6. Il canale fra l'erogatore e la gettoniera è dotato di una ventola che espelle le ultime possibili impurità attraverso i fori di cui il canale stesso è provvisto.
7. Inoltre l'Aventador è provvisto di un pulsante accessibile  che permette di eseguire a comando l'operazione di pulizia della catena. Tale pulsante si illumina in colore:
- verde fisso, quando l'apparecchio è in stand-by; il sistema è in attesa di attivazione; si può eseguirne la pulizia premendo il pulsante;
- giallo intermittente lento, quando il sistema è in fase di conteggio;
- giallo intermittente rapido, quando è in corso l'operazione di pulizia;
- rosso, se il sistema è fuori servizio.



1.2 Sicurezza

Il sistema Aventador deve essere installato all'interno di apparecchiature dotate di dispositivi per disconnettere l'alimentazione di rete.

Prima di togliere o montare l'Hopper Aventador, scollegare sempre l'alimentazione.

Il dispositivo contiene parti meccaniche in movimento: NON INTRODURRE le dita o oggetti diversi dalle monete mentre il dispositivo è in funzione, o quando la macchina è accesa.

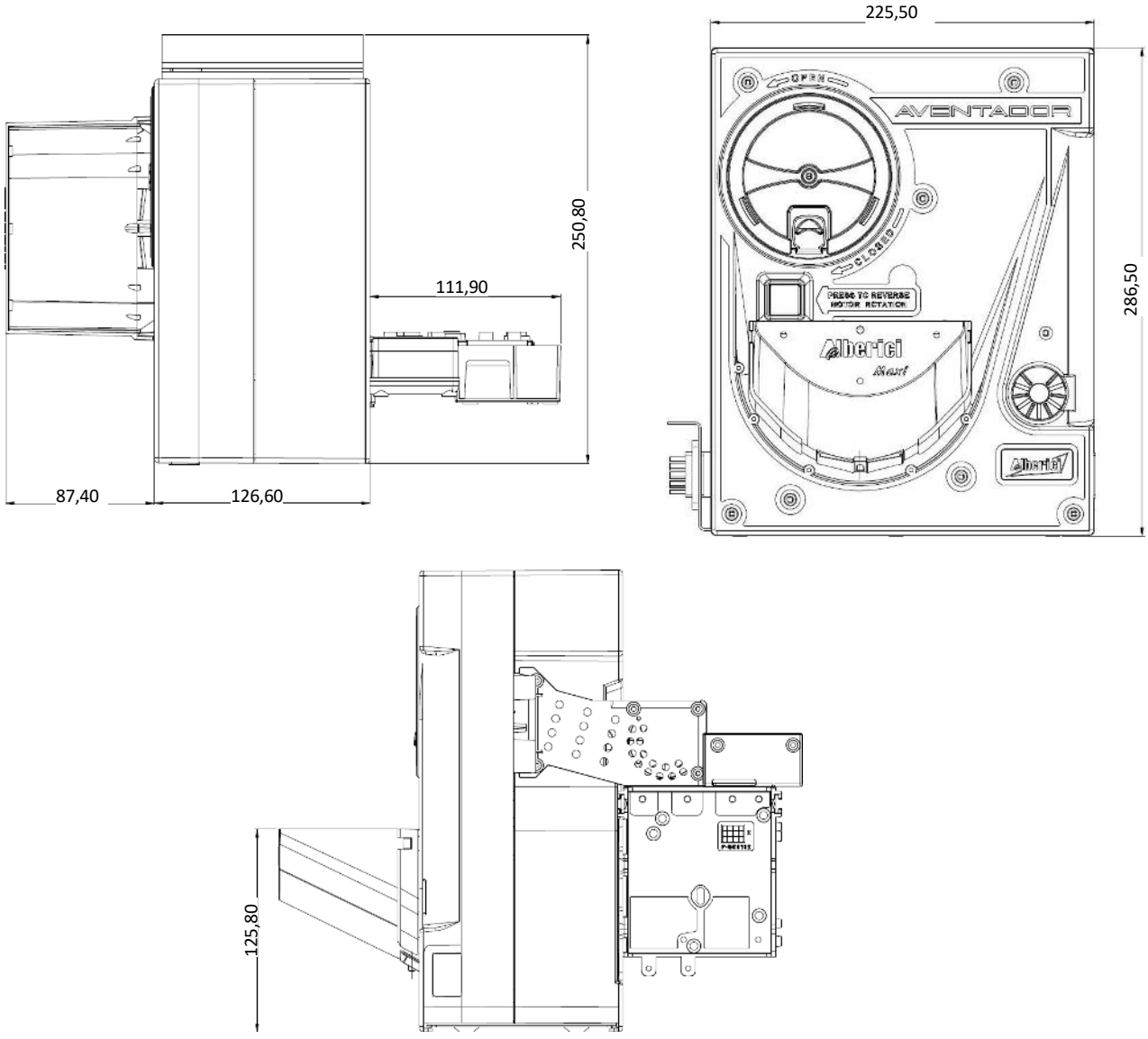
ATTENZIONE:
PERICOLO DI LESIONI!
MECHANICAL PARTS IN MOTION

2. Caratteristiche tecniche

Protocollo	***ccTalk 24V***
Velocità	***200 monete/min.***
Capacità monete (miste)	***400 miste***
Diametri accettati (mm)	***21 ÷ 32 mm (a richiesta: 16 ÷ 24 mm)***
Spessori accettati (mm)	***1,7 ÷ 2,4 mm***
Assorbimento Max	***1,40 A***
Assorbimento in Stand-by	***220 mA***
Tensione di funzionamento	***24 Vcc***
Temperatura di funzionamento	***0°C ÷ 75°C***
Umidità di riferimento	***20% ÷ 75% (non condensata)***
Peso (Kg)	***2,366***

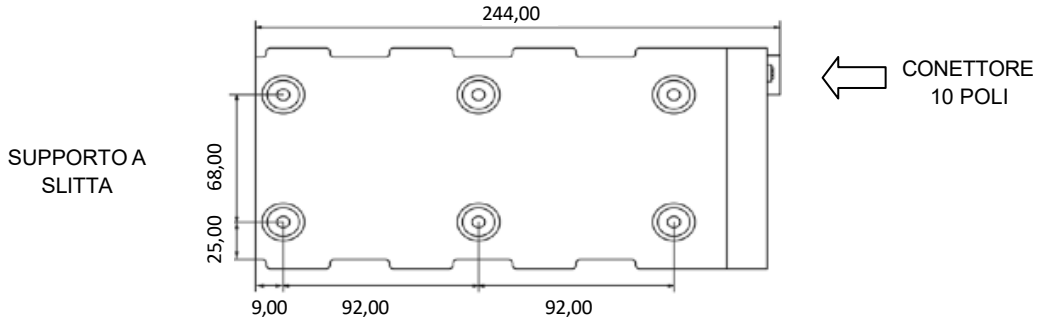
3. Descrizione meccanica

3.1 Dimensioni



3.2 Installazione

Eseguire le seguenti semplici operazioni:
. fissare il supporto a slitta al piano della macchina
. posizionare l'Aventador e traslarlo orizzontalmente fino al punto di battuta
. prima di collegare l'alimentazione, leggere il capitolo 4



Per smontare l'Aventador, tenere fermo il supporto, e slittare l'hopper verso l'esterno.
Eseguire l'installazione secondo le istruzioni del § 3.2, pena la decadenza della garanzia.

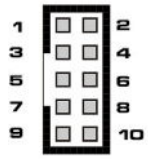
4. Descrizione elettrica

4.1 Alimentazione

L'alimentazione fornita all'Hopper Aventador deve essere a 24V in corrente continua. La sezione dei conduttori deve essere compatibile con l'assorbimento segnalato nella sezione 2.

4.2 PIN-OUT dei connettori

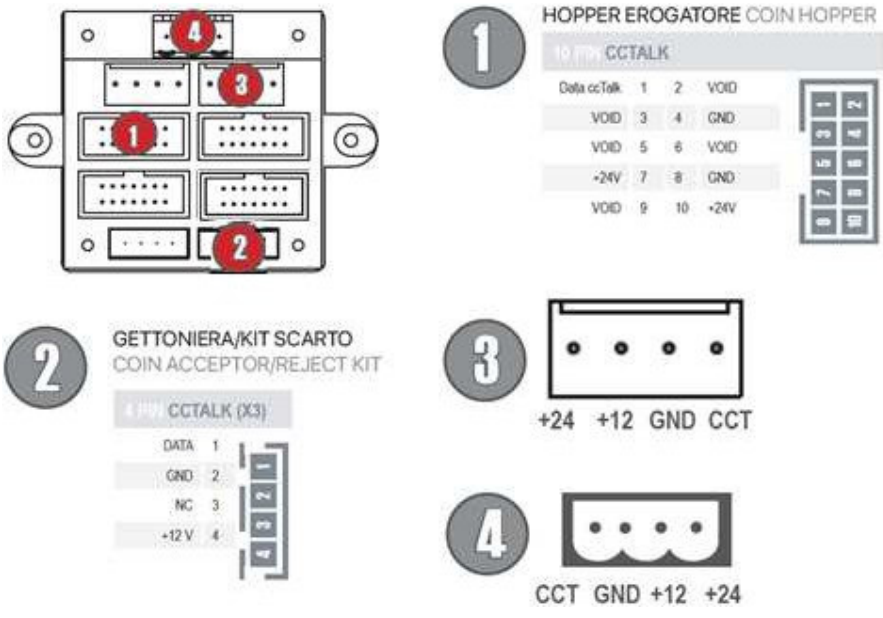
Sul retro dell'hopper si trovai connettore 10-pin ccTalk, accanto al banco di dip-switch per l'indirizzamento seriale. Tutti i segnali sono in logica negativa, ovvero il segnale è attivo quando il suo potenziale è uguale a GND.



1: DATA CCTALK
2: *VOID*
3: *VOID*
4: GND
5: *VOID*
6: *VOID*
7: +24V
8: GND
9: *VOID*
10: +24V

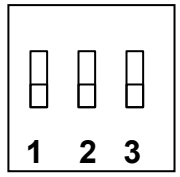
Si raccomanda di collegare tutte le lamelle presenti sull'hopper alla terra della macchina.

I dispositivi facenti parte del sistema sono collegati fra loro tramite il concentratore situato sul retro, secondo lo schema raffigurato qui sotto:



4.3 Predisporre l'indirizzo seriale tramite Dip-Switch

Quando necessario, l'indirizzo seriale dell'hopper Alberici può essere modificato via hardware mediante il banco di Dip-Switch posti sul retro. I 3 interruttori a slitta presenti possono essere combinati come da tabella seguente, per ottenere l'indirizzo conveniente:



(Switch n° 1) (Switch n° 2) (Switch n° 3)

Add. Sel 1	Add. Sel 2	Add. Sel 3	Serial Address
			3
ON			4
	ON		5
ON	ON		6
		ON	7
ON		ON	8
	ON	ON	9
ON	ON	ON	10

Tener presente che lo stato di questi dip-switch viene letto soltanto all'accensione o al reset, quindi lo spostamento durante il funzionamento non ha alcun effetto fino alla riaccensione.

5. Manutenzione

Prima di qualsiasi operazione di manutenzione che richieda lo smontaggio del sistema, spegnere l'alimentazione e staccare il cavo.
Controllare spesso, posizionandosi all'altezza della tramoggia di versamento delle monete, se sul fondo della catena dell'erogatore sono presenti detriti o corpi estranei, e rimuoverli (eventualmente utilizzare il getto di una bomboletta di aria compressa). La loro presenza può ostruire l'uscita delle monete, ostacolare il movimento della catena, guastare i componenti dell'Aventador o alterare le sue prestazioni.

E' buona norma attivare periodicamente il pulsante di pulizia automatica  : l'erogatore fa ruotare la catena in senso inverso, forzandone così la pulizia tramite la spazzola incorporata. Mantenere premuto il pulsante per il tempo necessario. E' anche utile soffiare aria compressa sulle finestrelle di scarico dei detriti e dei liquidi (cfr. immagine a pag. 4), per sbloccare eventuali otturazioni. Eseguire la stessa operazione sul canale traforato che si trova tra l'uscita dell'erogatore e l'ingresso della gettoniera.

Le monete arrivano nella gettoniera già abbondantemente trattate; tuttavia, nel tempo è possibile che nella sua camera di lettura si accumulino detriti. Alla camera di lettura si accede aprendo la basculante: innanzitutto rimuovere i detriti che si sono depositati tramite il getto di una bomboletta di aria compressa, e poi strofinare le pareti con un panno morbido senza pelacchi, leggermente imbevuto di detergente per vetri e superfici plastiche.

ATTENZIONE:

1) non usare alcohol, diluente, essenza di trementina, acetone, benzina o altri prodotti derivati da petrolio, o contenenti acido citrico.

2) le lenti dei sensori sono costituite di polimeri trasparenti, e vanno pulite con grande cautela per non graffiarle. Ripassare delicatamente varie volte, fino alla rimozione dello sporco.

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 Aventador is implemented according to generic specification 4.4

1 Communication specifications

ccTalk serial communication is derivation of RS232 standard.

Low data rate NRZ (***N**on **R**eturn to **Z**ero*) 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.

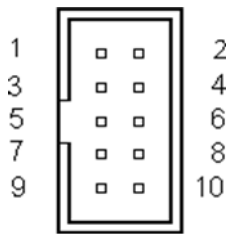


Fig. 1 - ccTalk connector lay-out

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

Table 1 - 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 2.

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

Table 2 Standard address for different types of devices

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** (***ack****nowledge*) replay of device to host.
Header 5 stands for **NAK** (***No ack****nowledge*) 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 (***B****inary* ***C****oded* ***D****ecimal*)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 recommendation.

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 Aventador Command header set

Command header set, that host could use in communication with Hopper is given in table 3.

Code / Hex.		Command header	Note
254	FE	Simple poll	Return ACK
253	FD	Address poll	MDCES support
252	FC	Address clash	MDCES support
246	F6	Request manufacturer id	"Alberici"
245	F5	Request equipment category id	"Payout"
244	F4	Request product code	"HopperTwo ccTalk"
242	F2	Request serial number	[Ser nr-L][Ser nr][Ser nr-H]
241	F1	Request software revision	un.n pm.m.m
217	D9	Request payout high/low status	Return empty/full status
210	D2	Modify sorter path	Supported: as many directions as allowed by the Sorter (2, 3, or 5)
197	C5	Calculate ROM checksum	[Mon][Prog][Data]
164	A4	Enable hopper	Enable = 0xA5
163	A3	Test hopper	Supported
134	86	Dispense hopper value	Supported
133	85	Request hopper polling value	Supported
132	84	Emergency stop value	Supported
131	83	Request hopper coin value	Supported
130	82	Request indexed hopper dispense count	Supported
1	1	Reset device	Software reset

Table 3 List of Hopper ccTalk command headers

Command headers are divided into 4 different groups:
- Common command headers
- Hopper command headers
- MDCES command headers

ALBERICI SPECIFIC COMMAND HEADERS – COMANDI SPECIFICI ALBERICI INERENTI IL SISTEMA AVENTADOR

Prima di entrare nello specifico elenco dei comandi, è utile la seguente descrizione che spiega come interagiscono tra di loro.

1) *All'hopper Aventador viene richiesta l'erogazione delle monete*
2) *La gettoniera viene interrogata per conoscere il tipo di moneta che ha ricevuto dall'hopper*
3) *Se la gettoniera va in errore, viene richiesta l'apertura e la ri-chiusura del suo sportello basculante, tramite il dispositivo cctalk apposito di sblocco automatico.*

I comandi specifici per l'hopper Aventador sono i seguenti:
242 - richiesta del numero di serie
164 - abilitazione
167 - erogazione
166 - polling (solo per capire se è fermo, non per interpretare il numero delle monete erogato)
163 - test hopper
172 - stop

I comandi per la gettoniera sono i seguenti:
231 - Abilitazione delle monete
210 - (Opzionale) impostazione del separatore
229 - Polling per capire il valore della moneta accettata e se ha qualche errore

I comandi per il dispositivo di sblocco (apertura della basculante) sono i seguenti:
240 - Apertura e chiusura basculante gettoniera

2.1 Comandi per l'hopper Aventador:

Default address = 03, salvo diversa configurazione del dip switch

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 format 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

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 (solo per capire se è fermo, non per interpretare il numero delle monete erogato)
This command return 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 Comandi per la gettoniera

Default address = 02

2.2.1 Command 231 [hexE7], Modify inhibit status

With this command host is able to inhibit the acceptance of some or all coins.
Acceptance or inhibition is set with a two byte mask sent by host.
Bits from 0 do 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 (opzionale: impostazione del separatore)

With this command the host is able to 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) represent scoin 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, and 5 way sorters with 3 coils. 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 value[1] 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

1 If coin selector use CVF (*Coin Value Format*)

128	*80*	*Inhibited coin (position 1)*	*Yes*
…	*…*	*Inhibited coin (position n)*	*Yes*
143	*8F*	*Inhibited coin (position 16)*	*Yes*
255	*FF*	*Unspecified alarm code*	*-*

Table 4 Acceptance error codes

Coin selectors also use one eight bit counter[2] that is incremented each time a new coin is detected. At the same time data in coin credit buffer are shifted two position 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 data[3]. Message format is:
Host sends: ***[Dir] [00] [00] [E5] [Chk]***
Coin sorter respond: ***[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][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][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][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 Comandi per il dispositivo di apertura e ri-chiusura della basculante.

Default address= "F0"

2.3.1 Command header 240 [hexF0], open/close coins selector door

Command format is
Host sends: ***[Dir][01][01] [FF] [11][Chk]***
Device answers: ***[01] [00] Dir] [00] [Chk] ACK***

2.4 Comandi rif. Algoritmo generico:

Lettura numero di serie dell'Aventador (importante perché altrimenti non possiamo farlo erogare) [242]

Abilitazione dell'hopper Aventador (importante perché sennò non possiamo farlo erogare) [164]

Abilitazione delle monete accettate dalla gettoniera (altrimenti scarterà tutto)

Impostazione della separazione delle monete accettate nel separatore [210]

Erogazione hopper Aventador (chiedere 250 monete) [134]

Ciclicamente, minimo ogni 500ms, ottimale 150ms:, si raccomanda di:
- *controllare lo stato di erogazione dell'hopper [166]*
- *controllare le monete accettate dalla gettoniera [229]*

Dal ciclo è possibile uscire in determinate condizioni:
- *se l'hopper ha erogato tutte le monete richieste, quindi dobbiamo ricomandare una nuova erogazione, perché il numero di monete che conteneva è piu alto del numero massimo che possiamo chiedergli in ciascun comando (interpretare il comando 166);*
- *se l'hopper si è fermato senza arrivare al numero di monete richieste. In tal caso dobbiamo capire se si è fermato per time-out (serbatoio vuoto) o per un errore. Per determinarlo usiamo il comando test hopper (interpretare il comando 163);*
- *se la gettoniera risponde un codice di errore. (Interpretare il comando 229) e in tal caso inviare il comando di stop dell'hopper (comando 172)*

2 Event counter
3 See command 249 Request polling priority

In caso di errori, si può attivare il meccanismo di scarto della moneta (comando FF-11)

IMPORTANTE: le monete accettate dal sistema devono essere quelle confermate dalla gettoniera al comando 229.

3.0 Setting the Hopper Address via PCB DIP-sw

The default address of Alberici hoppers can be changed by setting the onboard switches. The following table shows the possible Switch combinations 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

Table 5 Address selection for hoppers AH03-CD.

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

1 For details see ccTalk44-2.pdf, Address poll
2 252 bytes of data, source address, header and checksum (total of 255 bytes)
3 See Error handling
4 Refer to Appendix 3.1 of protocol document cctalk43-3.pdf

Alberici®
CASH SOLUTIONS

DICHIARAZIONE DI CONFORMITÀ

CE

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,**

D I C H I A R A

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

Modello	Configurazione	N° di Serie e/o matricola
AVENTADOR	❑ **CCTALK 24V**	_ _ _ _ _ _ _ _ _ _ _ _ _

Essendo realizzato conformemente al modello denominato AVENTADOR prototipo, finito di testare con esito positivo ai fini EMC e LVD (rapporto 7416-AVENTADOR.doc) il 20/02/2017, 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 direttive 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 61-226);
- CEI EN 60950-1 (CEI 74-2);
- CEI EN 61000-3-2 (CEI 110-31);
- 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 essenziali di sicurezza della Direttiva Bassa Tensione:
- L. 791 del 18 Ottobre 1977 e s.m.
- 2014/35/UE del 26 Febbraio 2014;

c) in conformità ai requisiti essenziali di sicurezza della Direttiva Compatibilità Elettromagnetica:
- D.Lgs. 194 del 06 Novembre 2007
- 2014/30/UE del 26 Febbraio 2014;

che conferiscono la presunzione di conformità alla Direttiva 2014/30/UE

Castel San Pietro Terme (BO), Italia lì, ___/___/_____

Fabrizio Alberici

Il Presidente

Alberici S.P.A.
Progettazione e produzione sistemi di pagamento, accessori per videogames e vending machines
Via Ca' Bianca 421, 40024 Castel San Pietro Terme (BO), Italia
Telefono:+39-(0)51-944300 r.a. – Fax:+39-(0)51-944594 – P.Iva:00627531205
***E-mail:** info@alberici.net – **Url:** http://www.alberici.net*

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.

Alberici

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

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

Tel.: +39.051.944300
Fax.: +39.051.944594

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