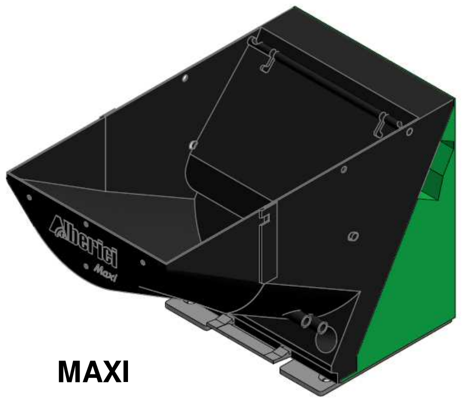
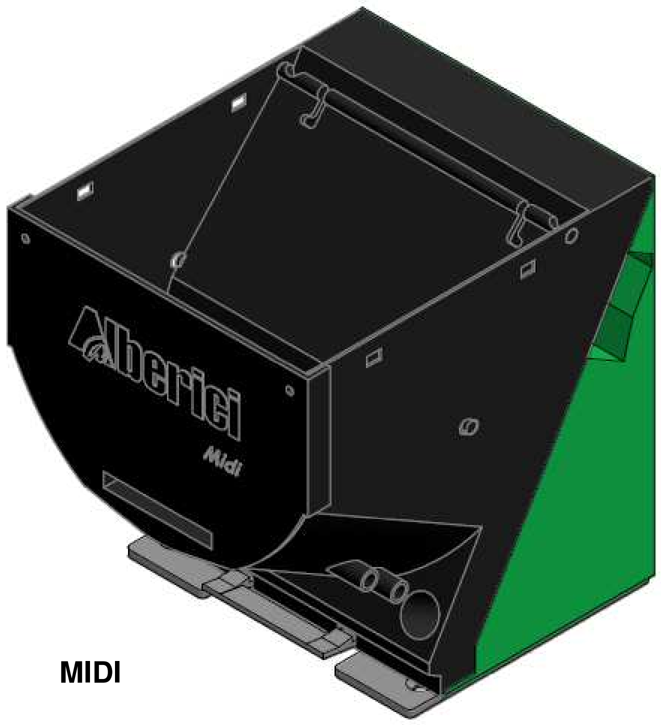
HopperCD EVOLUTION II

Mini - Midi - Maxi - Lateral

24Vdc ccTalk



MAXI



MIDI

MINII

LATERAL

Manuale d'uso

CE

Alberici®
CASH SOLUTIONS

Progettazione e produzione di sistemi di pagamento e accessori per macchine Gaming, Vending e Car-Wash

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

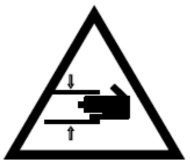
Congratulazioni per l'acquisto dell'**Hopper Evolution Alberici.**
L'Hopper Evolution è stato progettato nei laboratori e fabbricato nelle officine Alberici S.p.A., per soddisfare le più severe esigenze della gestione dei pagamenti.
Le tecnologie implementate lo rendono capace di gestire operazioni multiple, come identificare differenti valori di monete, contarle ed erogarle. Può essere corredato con un separatore per dividere le monete contate -ed eventualmente quelle rifiutate- in 2, o 3, o 5 direzioni diverse: rispettivamente separatore CD2, CD3, CD5.
E' indicato per l'impiego in **macchine Cambia-Cambia**.

1.1 Funzionamento

L' Hopper Evolution Alberici S.p.A. può riconoscere e contare vari tipi e valori di moneta o di gettoni (purché di forma circolare, senza solchi, senza fori o rilievi pronunciati), di diametro Φ 20 mm - 26,5 mm, e di spessore # 1,7 mm – 2,6 mm.
Man mano che eroga le monete, identifica e conta i tagli disponibili, e trasmette l'informazione via protocollo ccTalk all'host (scheda della macchina). Il riconoscimento di ogni moneta si ottiene mediante un dispositivo encoder utilizzato per la scansione di 4 differenti parametri magnetici delle monete, mentre una speciale leva con doppio sensore ne misura il diametro: la combinazione di questi dati permette l'accurata identificazione del valore.
La moneta in uscita interrompe momentaneamente il raggio infrarosso modulato fra un l.e.d. emettitore e un foto-transistor: ogni interruzione indica che è avvenuta l'erogazione di una moneta, il cui valore era stato subito prima individuato. La particolare cifratura del segnale di controllo impedisce tentativi di frode tramite accecamento del sensore.
Se si verifica un blocco temporaneo, l'elettronica dell'Hopper Evolution rileva il sovra-assorbimento di corrente, e inverte temporaneamente il senso di rotazione del disco, così da liberare l'uscita e ristabilire il normale flusso di erogazione.
Il livello delle monete nella tramoggia può essere controllato mediante sensori ottici che segnalano il superamento della quota di minimo.
La velocità di erogazione in condizioni standard è di ca. 180 monete al minuto.

1.2 Sicurezza

L'Hopper dovrà essere installato all'interno di sistemi o di apparecchiature che siano dotati di dispositivi di disconnessione dell'alimentazione di rete.
Prima di togliere o montare l'Hopper Evolution, staccare sempre l'alimentazione. Il dispositivo contiene parti meccaniche in movimento: NON INTRODURRE le dita o oggetti mentre è in funzione, o quando la macchina è accesa.



ATTENZIONE:
PERICOLO DI LESIONI!
MECHANICAL PARTS IN MOTION

Il disco dell'Hopper Evolution espelle le monete ad alta velocità: non posizionare parti sensibili del corpo o oggetti fragili di fronte alla feritoia di uscita delle monete.

2. Caratteristiche tecniche

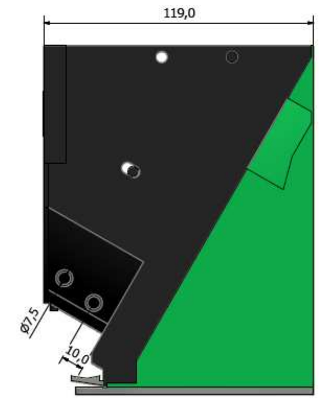
Protocollo	***ccTalk 24V***
Velocità	***180 monete/min.***
Capacità monete (Φ24 mm)	***775 / 725 / 500 / 325 (L / MAXI / MIDI / MINI)***
Diametri accettati	***20-26,5 mm***
Spessori accettati	***1,7-2,6 mm***
Assorbimento Max	***1 A***
Assorbimento* in *Stand-by	***70 mA***
Tensione di funzionamento	***24 Vcc***
Temperatura di funzionamento	***0°C ÷ 50°C***
Umidità di riferimento	***20% - 75%***
Dimensioni (mm)	***131 (l) x 154 (h) x 230 (p)***

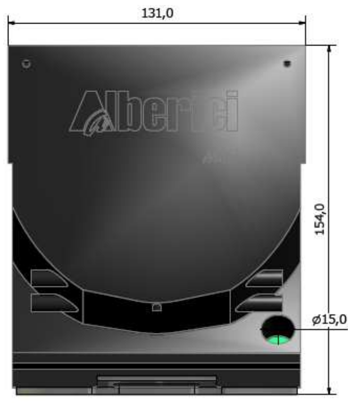
vers. SW: 6.0.8 al 08.05.2018

3. Descrizione meccanica

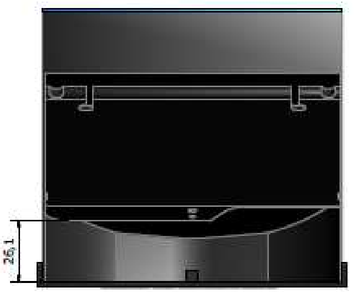
L'Hopper Evolution è disponibile in 4 modelli, differenti per capacità e dimensioni: Mini, Midi, Maxi, e Laterale. Le sue caratteristiche lo rendono intercambiabile con apparecchi analoghi sul mercato o già installati.

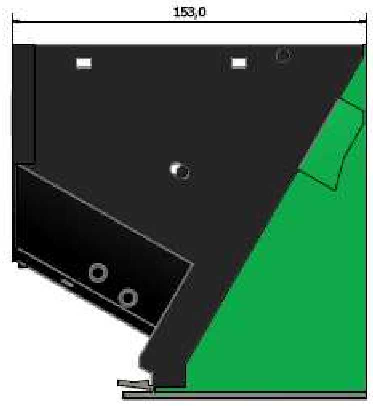
3.1 Dimensioni

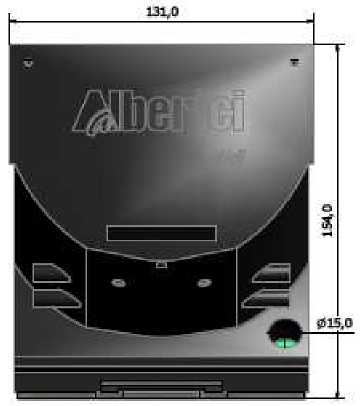


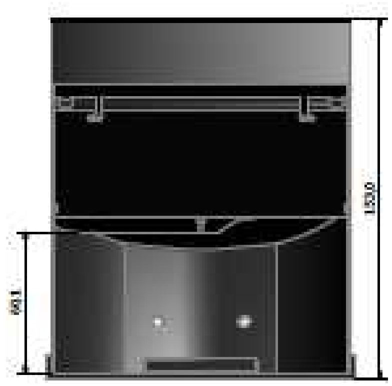


HOPPER EVOLUTION MINI

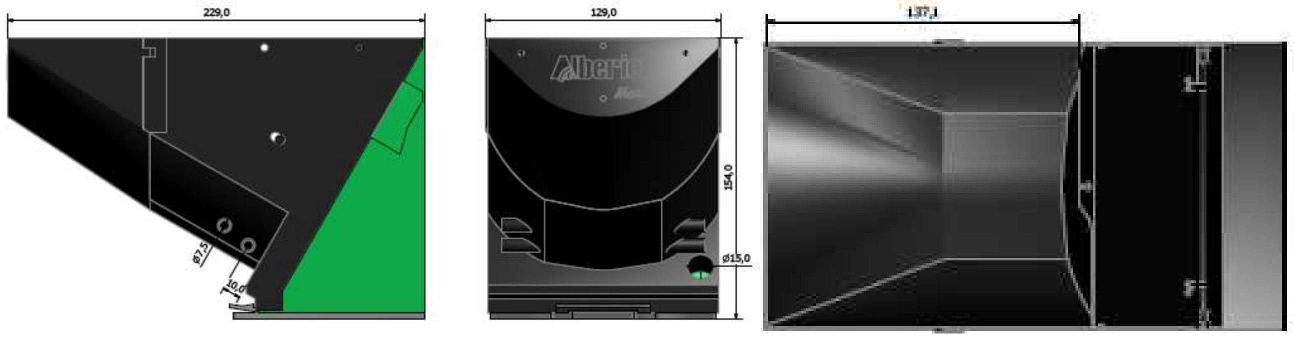




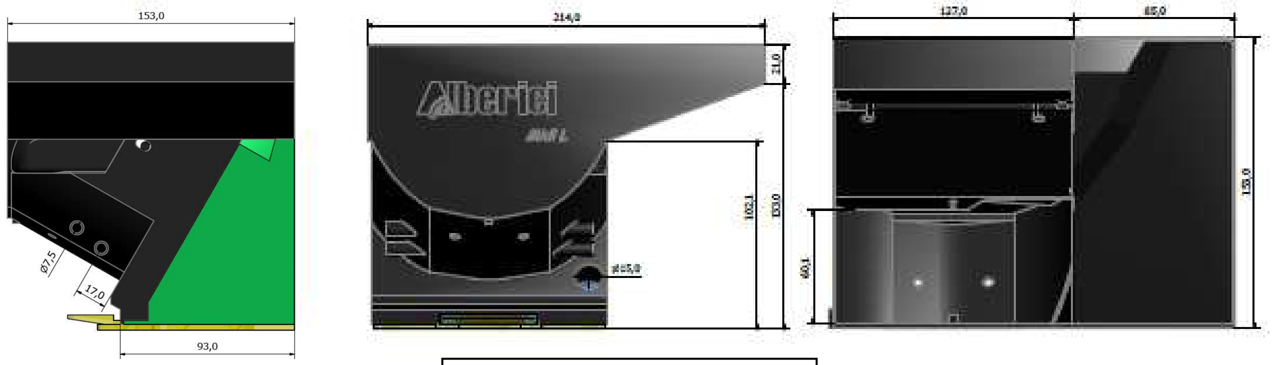




HOPPER EVOLUTION MIDI



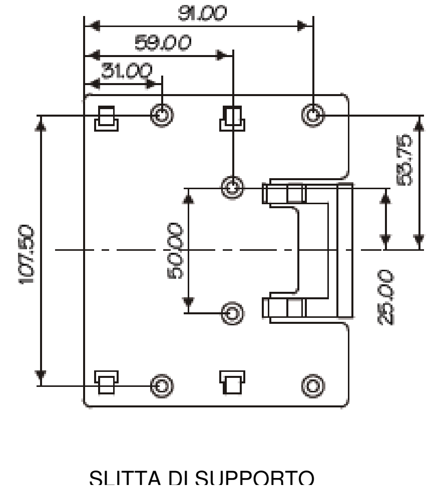
HOPPER EVOLUTION MAXI



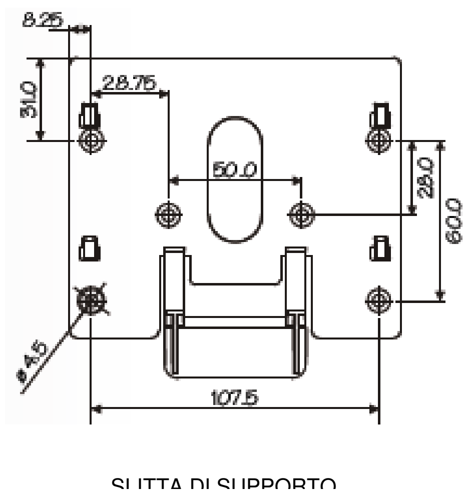
HOPPER EVOLUTION LATERAL

3.2 Installazione

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



SLITTA DI SUPPORTO



SLITTA DI SUPPORTO

Per smontare l'hopper, tenere premuta verso il basso la linguetta di sblocco del 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 Evolution deve essere a 24V in corrente continua. La sezione dei conduttori deve essere compatibile con gli assorbimenti segnalati di seguito.

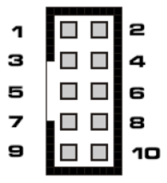
4.2 Assorbimenti

		Standby	**A Vuoto**	**Sotto carico (*)**
Scheda	**(+24 Vcc)**	**40mA 0.48 W**	**40mA 0.48 W**	**40mA 0.48 W**
Motore	**(+24 Vcc)**	**0mA 0m W**	**70mA 1.68 W**	**1 A * 24 W**
Totale		**0.48 W**	**2.16 W**	**24.48 W**

(*) L’assorbimento del motore sotto carico viene limitato elettronicamente. Il valore indicato viene raggiunto soltanto per una breve frazione di secondo *in caso di rotazione bloccata*.

4.3 PIN-OUT del connettore

Il connettore 10-pin ccTalk si trova sul retro dell’hopper, 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.

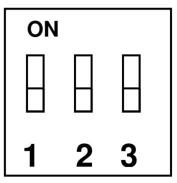


1: DATA CCTALK
2: *VOID*
3: *VOID*
4: GND
5: *VOID*
6: *VOID*
7: +24v
8: GND
9: *VOID*
10: +24v

In presenza di sensori di livello a tecnologia ottica, si raccomanda di collegare le lamelle comunque presenti sull’hopper alla terra della macchina.

4.4 Predisporre l’indirizzo seriale tramite Dip-Switch

Quando necessario, ad esempio quando si usa più di un Hopper Evolution sulla stessa macchina, 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**

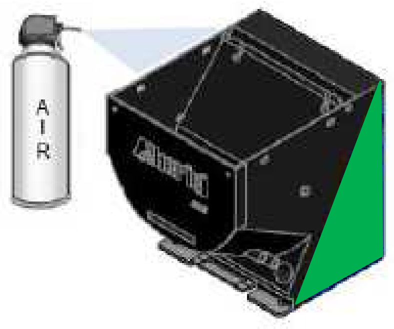
Tener presente che lo stato di questi switch viene letto soltanto all'accensione o al reset, quindi lo spostamento durante il funzionamento non ha alcun effetto fino alla riaccensione.

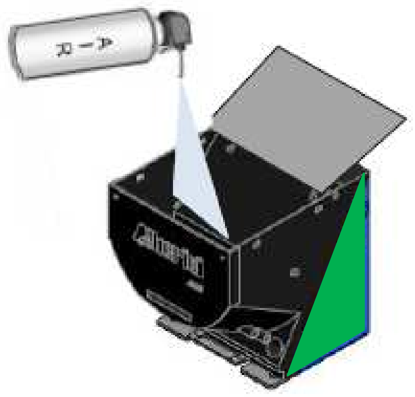
5. Manutenzione

Prima di qualsiasi operazione di manutenzione, spegnere l'alimentazione e staccare il cavo. Pulire il disco dell'Hopper Evolution almeno ogni 100.000 erogazioni, mediante un getto di aria compressa.

Controllare spesso, ad esempio in occasione di riempimenti manuali, che il disco o la tramoggia non contengano detriti o monete deformate, e rimuovere i corpi estranei senza indugio. La loro presenza può ostruire l'uscita, ostacolare la rotazione del disco, falsare la lettura dei valori, guastare i componenti dell'Hopper e rovinare le sue prestazioni.

Per pulire l'Hopper Evolution, sollevarne lo scivolo, e soffiare aria compressa non umida sui dischi, sulle finestrelle dei sensori (visibili attraverso i fori portamoneta), e attraverso la feritoia di erogazione.





E' comunque importante pulire periodicamente i dischi dell'hopper contatore e degli hopper erogatori (discriminatori), per evitare che l'accumulo di sporcizia possa falsare il corretto riconoscimento delle monete, provocando erogazioni errate in eccesso o in difetto.

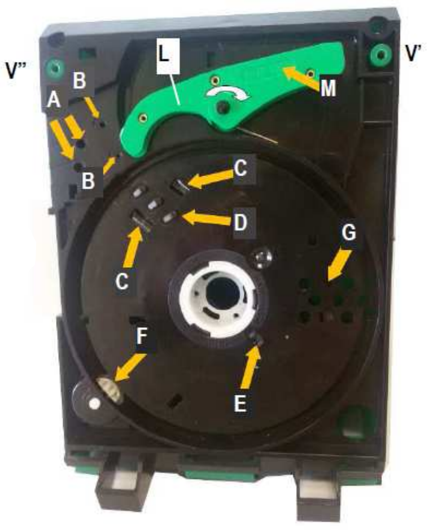
ATTENZIONE:

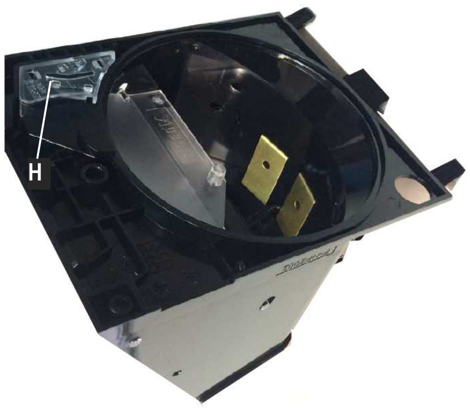
1) non usare alcohol, diluente, essenza di trementina, acetone, benzina o altri prodotti derivati da petrolio, o contenenti acido citrico.

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.

Eseguire la pulizia come segue:

1. Nel caso dell'Hopper contatore Evolution con separatore, staccare il cavo di collegamento tra i due e smontare il separatore (è tenuto da una sola vite).
2. Rimuovere le viti V' e V", ed estrarre la tramoggia dalla base triangolare, accedendo così ai dischi trascinatori.
3. Togliere i dischi e pulire il bordo dentato di quello nero. Rimuovere le impurità dalle superfici dei dischi, utilizzando uno straccio inumidito con una soluzione a base di Alcool Isopropile (no alcool denaturato, che potrebbe danneggiare i prismi dei sensori).
4. Ruotare la leva L di espulsione moneta, per scoprire i sensori situati dietro di essa nella posizione M: togliere la leva, e soffiare aria compressa nelle sedi dei sensori.
5. Pulire, sempre soffiando aria compressa, i sensori di accredito A e B. Pulire i prismi C e H.
6. Verificare la funzionalità dei dentini elastici D, e rimuovere eventuali detriti che ne intralcino il movimento.
7. Soffiare aria compressa sul sensore encoder E, e se necessario rimuovere lo sporco con un pennellino o uno scovolino morbido.
8. Soffiare aria compressa sul'ingranaggio conduttore F, e se necessario rimuovere lo sporco con un pennellino o con uno scovolino morbido.
9. Rimuovere eventuali detriti dalle feritoie G, in modo che le feritoie di drenaggio dei piccoli detriti introdotti con le monete restino efficienti.
10. Soffiare aria compressa sulla superficie del piatto, e completare la pulizia utilizzando uno straccio inumidito con una soluzione a base di Alcool Isopropile (non usare alcool denaturato, che potrebbe danneggiare i prismi dei sensori).
11. Riposizionare la tramoggia, e fissarla con le due viti V' e V".
    Nel caso dell'Hopper contatore Evolution, rimontare il separatore (posizionandolo sul lato e fissando la vite di serraggio), e riconnettere il cavo di collegamento all'hopper.





6. ccTalk protocol

6.1 ccTalk commands

NOTA: nella tabella sottostante sono elencati i comandi necessari per implementare l'Hopper Evolution II sulla scheda host.

Per visionare la lista dettagliata delle risposte dell'Hopper Evolution Alberici ai comandi dell'host (scheda macchina), cfr. § 6.2 ccTalk Protocol nella pag. seguente.

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	" AH EVM 1"
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 2 directions
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

6.2 cctalk Protocol of Hopper Evolution

HOPPER EVOLUTION - 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 Hoppers AH04.. 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

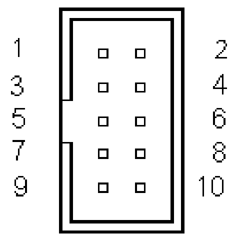
Data I/O line is "open collector" type, so it is possible to use device in systems with different voltage (*12V pull up in older devices*).

1.3 Connection

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

For schematics and and connector appearance see images and tables below.

Fig. 1 AH04 - ccTalk connector



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 AH04 - ccTalk connector

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 recommendation 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

Address for ALBERICI Hoppers is factory set to value 3, but the user can change the default address by setting the Hopper PCB switch.

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[2].

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** (***N****o* ***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[3].

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 time period of less than 250 ms.

During that 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 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	" AH EVM 1"
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, or 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 Evolution ccTalk command headers

Command headers are divided in to 4 different groups:

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

2.1 ALBERICI specific command headers

2.1.1 Command header 254 [hexFE], Simple poll

The fastest way for host to detect all attached devices in ccTalk network.

Addressed device - Hopper answer with ACK (*Acknowledge*).

If within predicted amount of time Hopper does not answer, probably is not connected, powered or simple not working properly.

Message format is:
```
 Host sends: [Dir][00][01][FE][Chk]
 Hopper answer: [01][00][Dir][00][Chk]
```

Hopper default address is 3, example of message packet is:
```
 Host sends: [03][00][01][FE][FE]
Hopper answer: [01][00][03][00][FC] ACK message
```

2.1.2 Command header 246 [hexF6], Request manufacturer ID

Hopper answer with ASCII string representing manufacturer name. In this case the hopper answer will be '**Alberici** '.

Message format is:
```
 Host sends: [Dir][00][01][F6][Chk]
Hopper answer: [01][Nr.b][Dir][00][a1][a2]...[an][Chk]
```

[Nr.b] is number of data bytes-characters sent by Hopper, and a1 to an are ASCII characters.
The example of message packet is:
```
 Host sends: [03][00][01][F6][06]
Hopper answer: [01][0E][03][00][41][6C][62][65][72][69][63][69][86]
```

2.1.3 Command header 245 [hexF5], Request equipment category ID

The answer to command header is standardized name for Hopper. Hopper will answer with ASCII string of characters representing standard name for that type of device '**Payout'**.

Message format is:
```
  Host sends: [Dir][00][01][F5][Chk]
Hopper answer: [01][06][Dir][00][50][61][79][6F][75][74][Chk]
```

Number of data byte is always 6, hex [06].

Example of message packets for hopper (*address 3*) is:
```
 Host sends: [03][00][01][F5][07]
Hopper answer: [01][06][03][00][50][61][79][6F][75][74][74]
```

2.1.4 Command header 244 [hexF4], Request product code

Hopper answer with ASCII string of character, representing its factory type.
For Alberici Hoppers it is '**AH EVM 1**'. Message format is:

```
 Host sends: [Dir][00][01][F4][Chk]
Hopper answer: [01][nr.byte][Dir][00][a1][a2]...[an][Chk]
```

Number of data bytes sent by Hopper is 9, hex [09].

Example of message packets for Hopper (*address 3*) is :
```
 Host sends: [03][00][01][F4][08]
 Hopper answer: [01][09][03][00][41][48][20][45][56][4D][20][31][12]
```

2.1.5 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][Ser.1-LSB][Ser.2][Ser.3-MSB][Chk]
```

The first data byte sent is LSB of serial number.

Example of message packets for Hopper (*address 3*) and serial number **1234567,** hex [BC][61][4E] is:
```
 Host sends: [03][00][01][F2][0A]
Hopper answer: [01][03][03][00][4E][61][BC][8E]
```

2.1.6 Command header 241 [hexF1], Request software revision

Hopper return ASCII string of character representing software version and revision. Message format is:
```
 Host sends: [Dir][00][01][F1][Chk]
Hopper answer: [01][Nr.b][Dir][00][a1][a2]...[an][Chk]
```

Number of data bytes in ASCII string is not limited and each producer has it's own system of labelling. Example of message packets for Hopper(*address 3*) is:
```
 Host sends: [03][00][01][F1][0B]
Hopper answer: [01][0B][03][00][75][31][2E][30][20][70][31][2E][30][2E][30][70]
```

In this case the Hopper answer is **'u1.0 p1.0.0'**. New generation of Hopper controllers has main firmware(program)FLASH up-grade capability built in small monitor program.

Monitor program version is marked with ASCII letter **'u'** and two digit's for minor and major changes. Main program is marked with letter **'p'** and three digit's for changes.

First digit is major program changes, second digit is for minor program changes and third is for error or bug corrections.

2.1.7 Command 197 [hexC5], Calculate ROM checksum

Hopper respond with three bytes of micro controller internal memory checksum.

First byte is monitor program FLASH checksum, second is main program FLASH checksum

and third is data(in RAM) checksum. Any changes in program memory or data will change the respond of hopper. Message format is:
```
 Host sends: [Dir][00][01][C5][Chk]
Hopper answer: [01][3][Dir][00][Cksum 1][Cksum 2][Cksum 3][Chk]
```

Example of message string for Hopper(*address 3*) is:
```
 Host sends: [03][00][01][C5][37]
Hopper answer: [01][03][03][00][53][6E][CC][6C]
```

2.1.8 Command header 1 [hex01], Reset device

After acceptance of command Reset hopper execute software reset and clear all variables in RAM or set them at the default value, including different counters, and any buffers. After reset hopper replay with ACK message.

Host software must re enable hopper to perform a new payout. Message format is:
```
 Host sends: [Dir][00][01][01][Chk]
Hopper answer: [01][00][Dir][00][Chk] ACK message
```

Example of message packets for hopper (*address 3*) is:
```
 Host sends: [03][00][01][01][FB]
Hopper answer: [01][00][03][00][FC] ACK message
```

Please pay attention: after reset, before getting able to start up again, the Hopper checks that the power supply is stable enough: it will not be available for use until the supply is acceptably stable. Depending on the power supply used, such time can vary from 500 msec up. Keep this in mind when assigning the timings for the next command instruction.

2.2 Hopper specific command headers

Hoppers are using some specific commands, for pay-out control, test of status and description. Some of commands are shared with other devices like banknote reader or hopper devices, but has different response or message format.

2.2.0 Command header 217 [hexD9],Request Payout Hi-Lo status

This command allow the reading of High/Low level sensor in payout systems.
Hopper answer with one byte that describe the sensors status.

The meaning of bits in status byte is the following:

Bit 0 - Low level sensor status.
0 – Higher than or equal to low level trigger
1 – Lower than low level trigger

Bit 1 – High level sensor status
0 - Lower than high level trigger
1 - Higher than or equal to high level trigger

Bit 4 - Low level sensor support
0 – Features not supported or fitted
1 - Features supported and fitted

Bit 5 - High level sensor support
0 - Features not supported or fitted
1 - Features supported and fitted

Bit's 2,3,6,7 are reserved bits

Message format is:
```
Host sends: [Dir][00][01][D9][Chk]
Hopper answer: [01][01][Dir][00][d1][Chk]
```

Example of message packets for Hopper(*address 3*) AH24 with no coins(*empty*) is:
```
 Host sends: [03][00][01][D9][23]
Hopper answer: [01][01][03][00][11][EA]
```

Only low sensor is supported, and hopper is empty.

2.2.1 Command header 164 [hexA4], Enable Hopper

This command enable hopper dispense. It must be sent once after power-on, Reset or Emergency stop command but before Dispense hopper coins command.

Message string format is:

```
 Host sends: [Dir][01][01][A4][d1][Chk]
 Hopper answer: [01][00][Dir][00][Chk] ACK
```

Data [d1] must be **hex [A5]** in order to enable hopper. Any other code will disable it.

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
```

2.2.2 Command header 163 [hexA3], Test Hopper

This command is used to test hopper hardware and report some problems in during dispense of coins. As response to that command hopper send to host a 2 bytes of data.

Each byte represent bit mask that show various hopper error. Bit meaning is shown below :

[Error mask 1]
Bit 0 – Absolute maximum current exceeded
Bit 1 – Payout time-out occurred
Bit 2 – Motor reverse during last payout to clear a jam
Bit 3 – Opto fraud attempt, path blocked during idle
Bit 4 – Opto fraud attempt, short circuit during idle
Bit 5 – Opto blocked permanently during payout
Bit 6 – Power up detected
Bit 7 – Payout disabled

[Error mask 2]
Bit 0 – Opto shorted during payout
Bit 1 – Flash data crc error
Bit 2 – Use other hopper to pay
Bit 3 – NU read 0
Bit 4 – Motor reverse limit end
Bit 5 – Unrecognized coin reverse limit
Bit 6 – Sorter blocked
Bit 7 – PIN mechanism active

Message string format is:

```
 Host sends: [Dir][00][01][A3][Chk]
 Hopper answer: [01][02][Dir][00][err 1][err 2][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][BA]
```

Such response is example of hopper state after power-up.

2.2.3 Command header 134 [hex86], Dispense hopper value

This command is used to dispense coin value from a discriminator hopper. The coin value is based as the lowest unit of coin(*1 cent, 1 pence etc.*). Maximum possible value to pay with one instruction is 65535(ie. cents or 655 Eur).

Hopper must be enabled before use of this command.

Message format is:
```
 Host sends: [Dir][05][01][86][sn1][sn2][sn3][Val-lo][Val-hi][Chk]
 Hopper answer: [01][00][Dir][00][Chk] ACK or NAK
```

Data string [sn1][sn2][sn3] is serial number, [Val-hi/lo] value to pay and [cnt] is number of pay out events!

First format message string example for hopper with ser. number (dec 1), to pay out 5 Euro is:
```
 Host sends: [03][05][01][86][01][00][00][F4][01][7B]
 Hopper answer: [01][01][03][00][FC] ACK
```

2.2.4 Command header 133 [hex85], Request hopper polling value

This command will return the value of four counters that are representing the state of current payment or last payment. These four counters are:

[Event Counter]

Each time a valid Dispense hopper coins command is received, event counter is incremented till it reach the value of 255. After that next pay out command will set this value to 1.
After power down or reset, value of event counter is 0.

[Payout value remaining]

This 2 byte counters will decrement for coin value after each coin is dispensed till it reach the value 0(both bytes) or stop of pay out. It shows us how much value is left to pay. Counter is set to new value after a valid dispense hopper value command.
It is cleared after Reset, after Emergency stop or after automatic pay stop.

[Last Payout: value paid]

This 2 byte counters will increment during the pay out with each dispensed coin.

The value of counter is saved in non-volatile FLASH memory in case if power failure occourr during the pay out cycle or after that. It is cleared at the begining of nex pay out comand. It shows us how much value has been paid out since last dispence command was lounched.

[Last Payout: value unpaid]

This 2 byte counters will also decrement during the pay out cycle in same way as counter

[Payout value remaining]. The difference is that this counter will be saved in non-volatile FLASH memory if power failure occourr during or after(*value 0*) pay out. It show us how much value was unpaid during last payout.

Message format is:
```
 Host sends: [Dir][00][01][85][Chk]
 Hopper answer: [01][07][Dir][00][d1][d2-l][d2-h][d3-l][d3-h][d4-l][d4-h][Chk]
```

Example of message packets for Hopper(*address 3*) is:
```
Host sends: [03][00][01][85][77]
Hopper answer: [01][07][03][00][01][00][00][00][00][96][00][5E]
```

In this example hopper didn't not perform a complete payout. The last payout was 3 Euro and 50 censts of 5 Euro to be paid.

2.2.5 Command header 132 [hex84], Emergency stop value.
This command immediately halt the payout sequence(*break the motor*) and reports back the value of coins which failed to be paid out. Hopper answer has 100 ms delay for last coin to exit. After Emergency stop value command, the hopper will be disabled.
To perform new payout sequence, hopper must be re-enabled.

Message format is:
```
 Host sends: [Dir][00][01][84][Chk]
Hopper answer: [01][02][Dir][00][d1-l][d1-h][Chk]
```

Example of message packets for Hopper (*address 3*) is
```
 Host sends: [03][00][01][84][78]
Hopper answer: [01][02][03][00][7E][04][78]
```

Data bytes hex[7E][04] represent hex 047E or dec 1150. That mean that there is 11 Euro and 50 cents left unpaid due to Emergency stop.

2.2.13 Command header 131 [hex83], Request hopper coin value.
This command return the "name" of specified coin as well as its value.
Coin name is standardized 6 byte ASCII characters[4]. It is possible to program different coin names that describe the one used in hopper. Unprogrammed coins has code:**'------'** .
This code is reserved for unknown(*any type*) of coin.

Message format is:
```
 Host sends: [Dir][01][01][83][01][Chk]
Hopper answer: [01][08][Dir][00][a1][a2][a3][a4][a5][a6][Val-lo]
[Val-hi][Chk]
```

Data a1 to a6 represent the coin "name"(*description*). Data Val-lo and hi are coin value.

Example of message packets for Hopper(*address 3*) coin nr. 1(*2 Euro*) is:
```
 Host sends: [03][01][01][83][77]
Hopper answer: [01][08][03][00][45][55][32][30][30][41][C8][00][BF]
```

Data bytes hex[45][55][32][30][30][41] are ASCII 'EU200A'.

Data bytes hex[C8][00] represent hex 00C8 or dec 200, the value of 2 Euro in cents.

2.2.14 Command header 130 [hex82], Request indexed hopper dispense count.

This command show the total number of each type of coins dispensed by hopper.
Each coin counter is non-volatile and has three bytes. LS byte is sent first.

Message format is:
```
 Host sends: [Dir][01][01][82][coin nr.][Chk]
Hopper answer: [01][03][Dir][00][d1][d2][d3][Chk]
```

Example of message packets for Hopper(*address 3*) and second coin is:
```
 Host sends: [03][00][01][82][02][54]
 Hopper answer: [01][03][03][00][54][00][00][A5]
```

In this example hopper dispensed 84 coins(*hex 54*) programmed on second channel.

Maximum value of dispensed coins stored in hopper FLASH is **16 777 215**.

2.2.15 Command header 210 [hexD2], Modify sorter paths

Host send one byte of coin position and one byte of sorter path. If sorter is present and path is correct(1 or 2) hopper will answer with **ACK**. Message format is:

```
 Host sends: [Dir][02][00][D1][Coin pos.][Sort.Path][Chk]
 Hopper answer: [01][00][Dir][00][Chk] ACK or NAK
```

Example of message string for hopper(*address 3*) to set sorter path 2 for coin position 1:

```
 Host sends: [03][02][00][D2][01][02][25]
 Hopper answer: [01][00][03][00][FC]
```

If sorter is not present or path is bigger than the number of paths allowed by the Sorter (2 paths, or 3 paths, or 5 paths), answer will be **NAK**.
Sorter path is initialy set to 1 and will return to that value after reset or power down.

2.3 MDCES command headers

MDCES stands for **M**ulti-**D**rop **C**ommand **E**xtension **S**et, or so called Multi-drop bus commands.

2.3.1 Command header 253 [hexFD], Address poll

This command is usually sent as a broadcast message by the host to determinate all address of device attached on ccTalk network. Hopper will answer with only one byte (*non-standard message format*), after a delay that is proportional to address value multiplied with 4 milliseconds.

Message format is:

```
 Host sends: [00][00][01][FD][Chk] Brodcast message
Hopper answer: Dly=4x[Address] -> [Address]
```

Example of message packets for Hopper (*address 3*) is:

```
 Host sends: [00][00][01][FD][02]
Hopper answer: Dly=12 ms -> [03] Address is 3
```

Example of message packets for Hopper (*address 250*) is:

```
 Host sends: [00][00][01][FD][02]
Hopper answer: Dly=1 s -> [FA] Address is 250
```

2.3.2 Command header 252 [hexFC], Address clash

Command Address clash has same replay format as address poll command, but time delay will be random. This will prevent the collision if two devices share same address.

Message format is:

```
 Host sends: [Dir][00][01][FC][Chk]
 Hopper answer: Random Dly -> [Address]
```

Example of message packets for Hopper (*address 3)* is:

```
 Host sends: [03][00][01][FC][00]
Hopper answer: Random Dly -> [03] Address is 3
```

3.0 Setting 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.

1For details see ccTalk44-2.pdf, Address poll
2252 bytes of data, source address, header and checksum (total of 255 bytes)
3See Error handling
4Refer to Appendix 3.1 of protocol document cctalk43-3.pdf

Alberici

DICHIARAZIONE DI CONFORMITÀ

CE

DIRETTIVA 2006/95/CE - DIRETTIVA 2004/108/CE

La ditta **Alberici S.p.A.**, avente sede in via **Guido Miglioli, 23, Zona Industriale, 40024 Castel San Pietro Terme (BO) – Italia,**

DICHIARA

Che il sistema classificato nella famiglia di prodotto **apparecchio elettrico d'uso domestico e similare – erogatore elettronico di monete**, identificato univocamente da:

Modello	Configurazione	N° di Serie e/o matricola
HOPPERCD EVOLUTION	☐ **CCTALK Mini** ☐ **CCTALK Midi** ☐ **CCTALK Maxi** ☐ **CCTALK Lateral**	____________

Essendo realizzato conformemente al modello campione testato con esito positivo ai fini EMC e LVD (rapporto 5500-HEVO.doc del 26 luglio 2010), dalla STP S.r.l., con sede legale in via Cervese, 373, 47521 Cesena (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:
- 2006/95/CE del 12 Dicembre 2006;
- L. 791 del 18 Ottobre 1977 e s.m.

c) in conformità ai requisiti essenziali di sicurezza della Direttiva Compatibilità Elettromagnetica:
- 2004/108/CE del 15 Dicembre 2004;
- D.Lgs. 194 del 06 Novembre 2007.

Che conferiscono la presunzione di conformità alla Direttiva 2004/108/CE

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

ALBERICI SPA Via Miglioli, [illegible] S. Pietro [illegible] (BO)

[illegible]

Il Presidente

Alberici S.P.A.
Progettazione e produzione sistemi di pagamento, accessori per videogames e vending machines
Via Guido Miglioli, 23, Zona industriale, 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