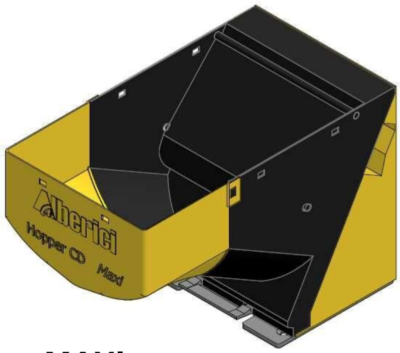
HopperCD ccTalk

Manuale d'uso

Rev. 1.04

HopperCD ccTalk

Mini, Midi, Maxi, Lateral

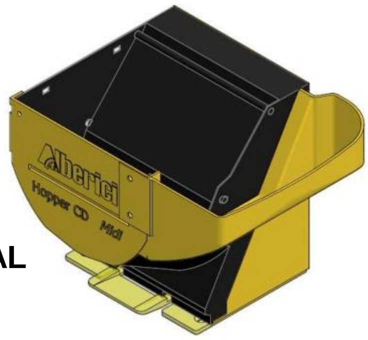


MAXI

MINI

MIDI

LATERAL



Manuale d'uso

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 DESCRIZIONE GENERALE

Congratulazioni per l’acquisto dell’HopperCD Alberici!

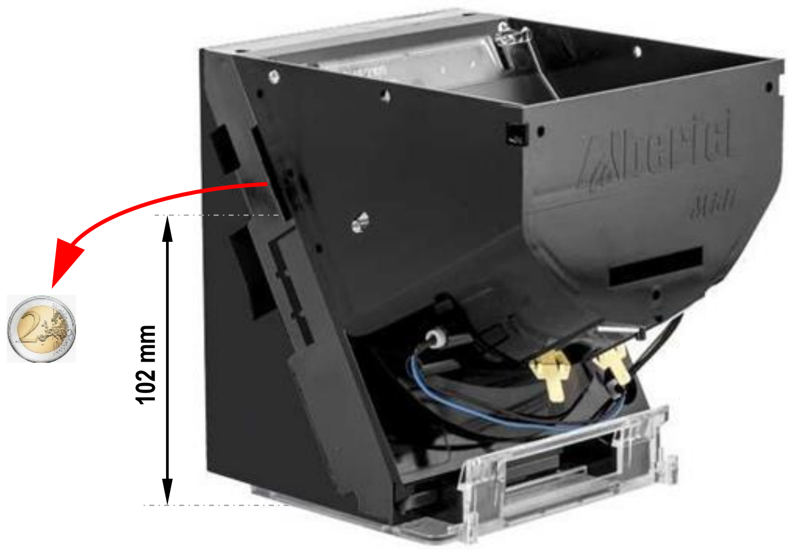
Siamo certi che apprezzerà la qualità e le prestazioni di questo apparecchio. Utilizza un disco raccoglitore e distributore, che rende rapida e costante l’erogazione delle monete. I componenti e le tecnologie utilizzate assicurano all’HopperCD affidabilità, precisione, e lunga vita utile. Il sistema funziona in protocollo cctalk, il noto standard di comunicazione seriale che garantisce precisione e sicurezza.

L’Hopper dovrà essere installato all’interno di sistemi o di apparecchiature che siano dotati di dispositivi di disconnessione dell’alimentazione di rete.

Collegare e scollegare l’HopperCD sempre con alimentazione spenta.

Non introdurre le dita nel dispositivo mentre è collegato all’alimentazione, e tantomeno durante il suo funzionamento: tale azione può provocare danni gravi, poiché all’interno sono presenti parti meccaniche in movimento e pilotate da un robusto motoriduttore elettrico.

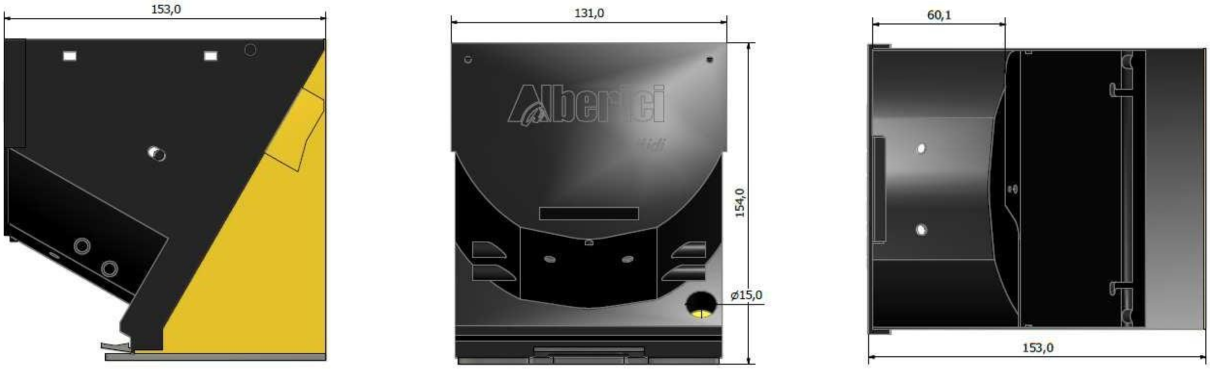
Protocolli	ccTalk, Standard Parallelo 24V
Velocità	280 mon./min
Capienza monete	775/725/500/325 (L/MAXI/MIDI/MINI)
Diametro monete	18-26,5 mm (disco a 7 fori) / 22-31,5 mm (disco a 6 fori)
Spessore monete	1,7- 2,6 mm
Assorbim. max	1 A
Assorbim. stand-by	40 mA
Alimentazione	motore 24 Vcc; ccTalk 24 Vcc
Temperatura lavoro	0°C ÷ 50°C
Umidità	20% - 75%
Dimensioni (mm)	Cfr. sezione 4



2 ASPETTI MECCANICI

E’ disponibile in quattro versioni di forme e capacità differenti: Mini, Midi, Maxi, Lateral. Si può presentare con base gialla o nera, a seconda del periodo di produzione.

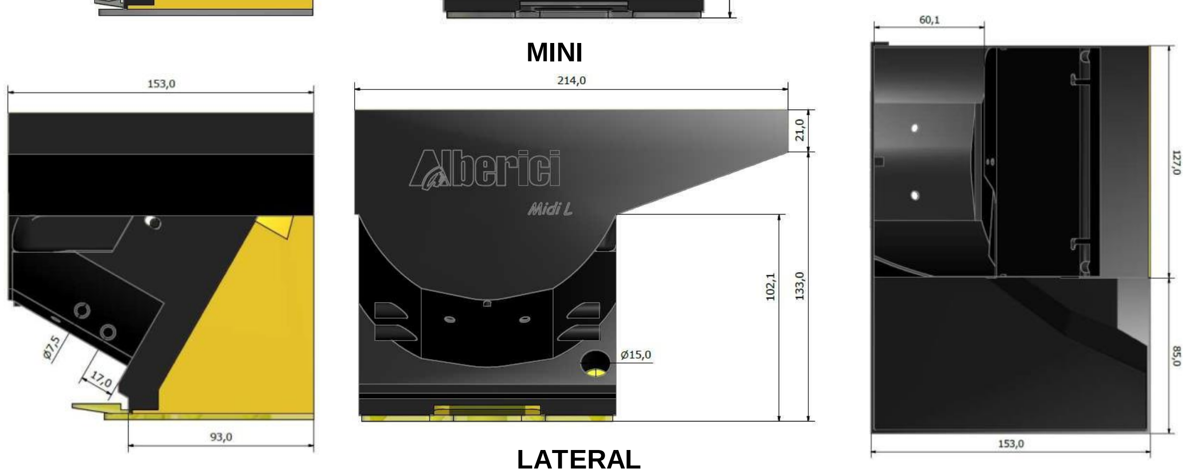
2.1 DIMENSIONI



MIDI

MAXI

MINI



LATERAL

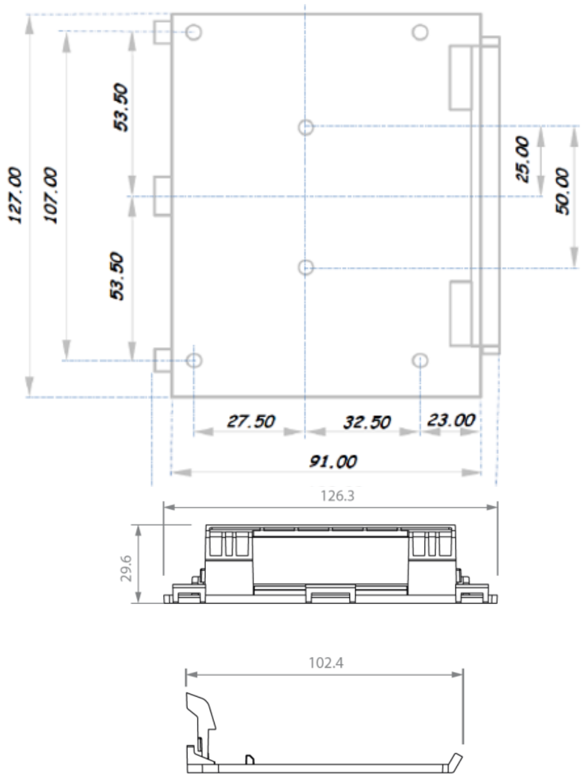
3 INSTALLAZIONE

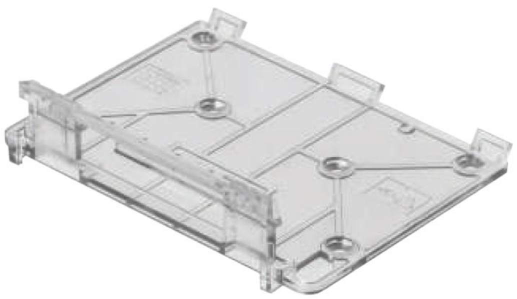
Assicurare la base in policarbonato sulla mensola di supporto dell'hopper all'interno del mobile.

Posizionare l'hopper sulla base, con il serbatoio verso la linguetta di sgancio, e spingerlo verso il basso.

Prima di collegare il cavo dell'alimentazione alla presa 10p sul retro dell'hopper, cfr. il capitolo 4.

Per rimuovere l'hopper, tenere tirata verso l'esterno la linguetta di sgancio e tirare l'hopper verso l'alto.





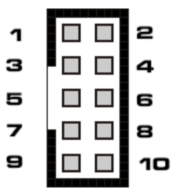
4 ASPETTI ELETTRICI

4.1 ALIMENTAZIONE

L'alimentazione fornita all'HopperCD deve essere a 24V in corrente continua. La sezione dei conduttori deve essere compatibile con gli assorbimenti sotto segnalati.

4.2 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.

4.3 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*.

4.4 BANCO INDIRIZZI

Quando necessario, ad esempio quando si usa più di un HopperCD 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'inditrizzo conveniente:

ON

1 2 3

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

Add. Sel 1	Add. Sel 2	Add. Sel 3	Indirizzo Seriale
			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'HopperCD almeno ogni 100.000 erogazioni, mediante un getto di aria compressa.

La conformazione della sede del sensore d'uscita impedisce l'accumularsi dello sporco, il che riduce la necessità di pulizie frequenti. E' comunque consigliabile pulire anche il sensore quando si pulisce il disco.

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'HopperCD, 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.
Per qualunque operazione di pulizia o di manutenzione che richieda lo smontaggio di particolari, si raccomanda di spedire l'HopperCD alla Alberici S.p.A., che provvederà a pulirlo e ad eseguire anche gli eventuali aggiornamenti necessari.

6 COMANDI PROTOCOLLO CCTALK

Communication ccTalk protocol

cctalk® communication protocol is the Money Controls[1] serial communiction 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.

The communication protocol of the Alberici ccTalk HopperCD is implemented according to generic specification 4.2

1 Communication specifications

Serial communication was derivated from RS232 standard.
Low data rate NRZ (***N****on* ***R****eturn to* ***Z****ero*) asyncronous 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 suported.
Message integrity is controled by means of checksum calculation.

1.1 Baud rate

The baud rate of 9600 was chosen as compromise betwen 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.

1.3 Connection

The connection of HopperCD at network is achieved by means of its 10-pin connector . Connector is used for power supply and for communication as well.
For schematics and and connector appearance see picture at page 4.

[1] Formally Coin Controls

1.4 Message structure

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

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

There is an exeption of message structure when device answer to instruction Address poll and Address clash[2]. The answer consists of only one byte representing address delayed for time proportional to address value. 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 caled "brodcast" address and address 1 is default host address.
The recomandations for address value of different devices are presented in table 1.

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 1 Standard address for different types of devices

Address for Alberici HopperCDc is factory set at value 3, but the user can change the default address using MDCES instructions Address change or Address random or setting Hopper external switch.

[2] For details see cctalk42-2.pdf, Address poll

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 lenght 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[3].

1.4.3 Command headers (*Instructions*)

Total amount of possible cctalk command header is 255 with possibility to add sub-heaers 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 are explained later on, for each specific message transfer.
Commands are devided 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 mechs
- MDCES commands

Alberici HopperCD use 24[4] instructions-headers. Details of all instruction use are explained in chapter 2.

1.4.4 Data

There is no restrictions data formats use. Data could be BCD (***B**inary **C**oded **D**ecimal*)numbers, Hex numbers or ASCII strings. Intrepretation 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 recieved and the addition of all bytes are non-zero then an error has occurred[5].
For noisy enviroment or higher security application it is possible to use more complex, 16 bit CRC CCITT checksum based on a polynomial of:
$\mathbf{x^{16} + x^{12} + x^{5} + 1}$ and initial value of CRC register **0x0000**.

Hopper are using simple checksum, but they could be set to operate with CRC-16 checksum on customer demand.

[3] 252 bytes of data, source address, header and checksum (total of 255 bytes)
[4] First level of implementation
[5] See Error handling

1.5 Timing specification

The timing requirements of cctalk are not very critical but there are some recomandations.

1.5.1 Time beetwen two bytes

When reciving bytes within a message packet, the comunication 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 recieve next message. The interbyte delay during transmition should be ideally **less than 2 ms** and **not greater than 10 ms**.

1.5.2 Time beetwen comand and replay

The time beetwen comand and reply is dependent on application specific for each comand. Some comands return data imediatly, and maximum time delay should be within **10 ms**.Others comands that must activate some actions in device may return reply after the action is finished

1.5.3 Start-up time

After the power-up sequence HopperCD 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 HopperCD should be able works fine.

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. HopperCD Command header set

Code		Command header	Note
254	FE	Simple poll	Return ACK
253	FD	Address poll	MDCES support
252	FC	Address clash	MDCES support
251	FB	Address change	MDCES support, non volatile
250	FA	Address random	MDCES support, non volatile
246	F6	Request manufacturer id	’Alberici group’
245	F5	Request equipment category id	‘Payout’
244	F4	Request product code	‘HopperTwo ccTalk’
242	F2	Request serial number	From 0 to 16.777.215
241	F1	Request software revision	‘X.xx’
219	DB	Enter new PIN number	Supported, non volatile
218	DA	Enter PIN number	ACK return if PIN is correct
217	D9	Request payout high/low stat.	Return empty/full status
216	D8	Request data storage availability	[00][00][00][00][00] ,not available
192	C0	Request build code	‘ALH02v00’
172	AC	Emergency stop	Return ACK
169	A9	Request address mode	[B7] add.changed with serial command(nv)
168	A8	Request hopp.dispense count	From 0 to 16.777.215
167	A7	Dispense hopper coins	Data = Serial number + N° of coin to disp.
166	A6	Request hopper status	Return dispensed coin counters
164	A4	Enable hopper	Data must be A7
163	A3	Test hopper	Return hardware status
4	4	Request comms revision	[02][04][02] ,level2, isue4.2
1	1	Reset device	Software reset

Table 2 List of Hopper cctalk command header

Command header set, that host could use in communication with Hopper is given in the table 2.

Command headers are divided in to 3 different groups:

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

2.1 Common command headers

Common commands are used in all type of devices to detect there presence on cctalk network or to describe them. Information like: manufacturer or product type id, serial number, different settings etc. are transmitted to host.

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.
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. For **Alberici group** Hopper, 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] [20] [67] [72] [6F] [75] [70] [86]**

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

Answer to command header is standardized name for Hopper. It answer with ASCII string of characters representing standardized 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 coin selector (*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 Hopper it's **HopperTwo ccTalk**. Message format is:

Host sends: **[Dir] [00] [01] [F4] [Chk]**

Hopper answer: **[01] [10] [Dir] [00] [a1][a2] . . . [an] [Chk]**

Number of data bytes sent by Hopper is 16, hex [10].
Example of message packets for Hopper (*address 3*) is :

Host sends: **[03] [00] [01] [F4] [08]**

Hopper answer: **[01][10][03][00][48][6F][70][65][72][54][77][6F][20][63][63][54][61][6C] [6B][D2]**

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] [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: **[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] [04] [03] [00] [31] [2E] [32] [31] [36]**

Hopper answer is '**1.21'**.

2.1.7 Command header 192 [hexC0], Request build code

Hopper answer with ASCII string of character representing it's hardware version and revision[6]. Last revision of printed circuit board for Hopper is **ALH02v00**.
Message format is:

Host sends:	**[Dir] [00] [01] [C0] [Chk]**
Hopper answer:	**[01] [Nr.b] [Dir] [00] [a1] [a2]............ [an] [Chk]**

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

Host sends:	**[03] [00] [01] [C0] [3C]**
Hopper answer:	**[01] [08] [03] [00 [41] [4C] [48] [30] [32] [76] [30] [30] [E7]**

2.1.8 Command header 169 [hexA9], Request address mode

Hopper answer with one data byte[7] information about address mode and options. Address could be stored in different type of memory (*RAM. ROM or EEPROM).* Some devices support address change wit MDCES command headers[8]. Message format is:

Host sends:	**[Dir] [00] [01] [A9] [Chk]**
Hopper answer:	**[01] [01] [Dir] [00] [Address mode] [Chk]**

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

Host sends:	**[03] [00] [01] [A9] [53]**
Hopper answer:	[01] [01] [03] [00] [B7] [44]

Hopper answer with data [B7]. It means that address may be changed with serial command (non volatile).If answer is [B3], mean that address is selected via interface connector.

2.1.9 Command header 4 [hex04], Request comms revision

Hopper answer with three byte data information about level of cctalk protocol implementation, major and minor revision. Message format is:

Host sends:	**[Dir] [00] [01] [04] [Chk]**
Hopper answer:	**[01] [03] [Dir] [00] [Level] [Mag.rev.] [min. rev.] [Chk]**

Example of message packets for Hopper (*address 3*), cctalk protocol issue **4.2**, is:

Host sends:	**[03] [00] [01] [04] [F8]**
Hopper answer:	**[01] [03] [03] [00] [01][04][02] [F2]**

2.1.10 Command header 1 [hex01], Reset device

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

[6] Usually label printed on electronic circuit board
[7] Details of description see in public document cctalk42-2.pdf
[8] Address change, Address random

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*) **AL06V-c** is:

Host sends:	**[03] [00] [01] [01] [FB]**	
Hopper answer:	**[01] [00] [03] [00] [FC]**	ACK message

2.2 Hopper command headers

Hopper use some specific commands, for paying or read itself status.
Some of commands are shared with other device like banknote reader or coin selector devices.

2.2.0 Command header 219 [hexDB], Enter new PIN number

Host send four byte data of new PIN number. If correct PIN was previously received[9] Hopper will accept the new PIN and answer with ACK message . Hopper has PIN number stored in EEPROM. Message format is:

Host sends:	**[Dir] [04] [01] [DB] [PIN1-LSB][PIN2][PIN3][PIN4-MSB] [Chk]**	
Hopper answer:	**[01] [00] [03] [00] [FC]**	ACK if PIN is correct
Hopper answer:	no answer if PIN is incorrect or not recieved	

Example of message packets for Hopper (*address 3*), with default PIN, hex**[00][00][00][00]** previously received and NEW pin hex**[01][02][03][04]** is:

Host sends:	**[03] [04] [01] [DB] [01][02][03][04] [13]**	
Hopper answer:	**[01] [00] [03] [00] [FC]**	ACK message

2.2.1 Command header 218 [hexDA], Enter PIN number

Host send four byte data of PIN number. If PIN is correct, Hopper will answer immediately with ACK message. If PIN is incorrect the NAK message will be sent with time delay of 100 ms. Hopper has PIN number stored in EEPROM. Message format is:

Host sends:	**[Dir] [04] [01] [DA] [PIN1-LSB][PIN2][PIN3][PIN4-MSB] [Chk]**	
Hopper answer:	**[01] [00] [Dir] [00] [Chk]**	ACK if PIN is correct
Hopper answer:	**[01] [00] [Dir] [05] [Chk]** dly 100 ms ->NAK if PIN is incorrect	

Example of message packets for Hopper (*address 3*), with default PIN, hex**[00][00][00][00]** and wrong pin is:

Host sends:	**[03] [04] [01] [DA] [01][00][00][00] [1E]**
Hopper answer:	**[01] [00] [03] [05] [F7]** dly 100 ms ->NAK if PIN is incorrect

[9] See next chapter

2.2.2 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 that byte is the following:

BIT0 - Low level sensor status.
0 – Higher than or equal to low level trigger 1 – Lower than low level trigger
BIT1 – High level sensor status 0 - Lower than high level trigger
1 - Higher than or equal to high level trigger

BIT4 - 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

BIT2,3,6,7 are reserved bits
Trigger level is set by fixed sensor into hopper mechanism.

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*) is

Host sends:	**[03] [00] [01] [D9] [23]**
Hopper answer:	**[01] [01] [03] [00] [31] [CA]**

Data byte Hex[31] mean that Hopper high and low sensor are supported, and hopper is empty.

2.2.3 Command header 216 [hexD8], Request data storage availability

Hopper answer with five byte of data that describes type of memory and availability for host to read and to write. Message format is:

Host sends:	**[Dir] [00] [01] [D8] [Chk]**
Hopper answer:	**[01] [05] [Dir] [00] [d1][d2][d3][d4][d5] [Chk]**

Alberici Hopper, at the moment, does not support write or read to memory. Answer to command is always as in example:

Host sends:	**[03] [00] [01] [D8] [24]**
Hopper answer:	**[01] [05] [03] [00] [00][00][00][00][00] [F7]**

2.2.4 Command header 172 [hexAC], Emergency stop.

This command immediately halts the payout sequence and reports back the number of coins which failed to be paid 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.

2.2.5 Command header 168 [hexA8], Request hopper dispense count.

This command show the total number of coin dispensed by hopper. Message

format is:

Host sends:	**[Dir] [00] [01] [A8] [Chk]**
Hopper answer:	**[01] [03] [Dir] [00] [d1] [d2] [d3] [Chk]**

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

Host sends:	**[03] [00] [01] [A8] [54]**
Hopper answer:	**[01] [03] [03] [03] [54] [00] [00] [A5]**

In this example hopper dispensed 84 coins (deicmal of Hex 54).

Maximum value of dispensed coin stored in hopper EEPROM is 16’777’215 (3Bytes).

2.2.6 Command header 167 [hexA7], Dispense hopper coin

This command dispense coin from the hopper. Maximum number of coin hopper can dispense with a single command is 255.
Before Dispense hopper coin command, hopper need to be enabled, else dispense action is not performed.

Alberici hopper answer correctly to two format of dispense coin command. First message format is

Host sends:	**[Dir] [04] [01] [A7] [sn1] [sn2] [sn3] [N°Coin][Chk]**	
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 try 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 answer:	**[01] [00] [03] [00] [FC]**	**ACK**

One token is paid.

2.2.7 Command header 166 [hexA6], Request hopper status

This command return four counters that explain the status of payment. These
four bytes are:
Event Counter that show the number of good dispense events since last reset.
Payout coins remaining that show how many coins are still to pay.
Last Payout: coins paid, that show how many coins paid out since last dispence command (increments with each coin dispensed)
Last Payout: coins unpaid, that show how many coins was 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 11^{th} payout before last reset. A coin is paid (9 are remaining) and during last payout 2 coin was paid.

2.2.8 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**

2.2.9 Command header 163 [hexA3], Test Hopper

This command is used to test hopper hardware. It reports back a bit mask that show various hopper error. 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] mean that Opto are blocked permanently during payout and Power up was detected.

2.3 MDCES command headers

MDCES stands for **M**ulti-**D**rop **C**ommand **E**xtension **S**et, or so called Multi-drop buss commands. Multi-drop buss commands gives additional functionality to systems that require change of address for devices in cctalk network.
Some of commands has different message format than usual cctalk message.
Commands are:

- Address poll
- Address clash
- Address change
- Address random

Because host always use address 1 and address 0 is for broadcast message all commands that changes the address should not accept this settings.

All changes are stored in non-volatile memory, EEPROM !

2.3.1 Command header 253 [hexFD], Address poll

This is a broadcast message used by host to determinate all address of device attached on cctalk network. Hopper 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 -> [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 answer from Hopper, like address poll command, but host issue this command with specific device address and not using broadcast address. Hopper answer with only one byte (*non-standard message format*), after a random value of time delay to prevent 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

2.3.3 Command header 251 [hexFB], Address change

Command Address change is issued to a specified device only. Hopper answer with ACK message. Message format is:

Host sends:	**[Dir] [01] [01] [FB] [Address] [Chk]**	
Hopper answer:	[01] [00] [03] [00] [FC]	ACK

Example of message packets for Hopper (*address 3*) and change in to address 20:

Host sends:	**[03] [01] [01] [FB] [14] [EC]**		
Hopper answer:	**[01] [00] [03] [00] [FC]**	ACK	Address is now 20

Hopper does not answer to attempt of change an address to 0 or 1.

2.3.4 Command header 250 [hexFA], Address random

Command Address random has the same answer from coin selector. New address is not sent because each device set its own random address.

Host software sometime can issue this command as broadcast. This will cause change of all device addresses. Hopper answer with ACK message.
Message format is:

Host sends:	**[Dir] [00] [01] [FA] [Chk]**	
Hopper answer:	**[01] [00] [03] [00] [FC]**	ACK

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

Host sends:	**[03] [00] [01] [FA] [02]**		
Hopper answer:	**[01] [00] [03] [00] [FC]**	ACK	Address is changed

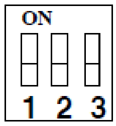
Example of broadcast message packets for Hopper is:

Host sends:	**[00] [00] [01] [FA] [05]** Brodcast message		
Hopper answer:	**[01] [00] [00] [00] [FD]**	ACK	Address is changed

Hopper has internal mechanism that prevent setting of address 0 or 1.

3.0 Setting Hopper Address via Hardware

Alberici hopper can change its default address via hardware, by positioning accordingly the 3 dip-swiches provided.



The following chart shows the possible combinations of signals with which you can determine the different address of the Hopper**.**

Add. Sel 1 (Switch n°3)	**Add. Sel 2** (Switch n°2)	**Add. Sel 3** (Switch n°1)	**Serial Address**
			3
		X	**4**
	X		**5**
	X	**X**	**6**
X			**7**
X		**X**	**8**
X	**X**		**9**
X	**X**	**X**	**10**

When the dip-switch is ON, it connects automatically the signal to Vs. Note that the hopper reads the status of these lines only at reset, so dip switch changes has no effect on its address during normal hopper working.

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 – erogatore elettronico di monete**, identificato univocamente da:

Modello	Configurazione	Tipo comunicazione	N° di Serie e/o matricola
HOPPER CD	☐ **Mini** ☐ **Midi** ☐ **Maxi** ☐ **Lateral**	☐ **CCTALK** ☐ **IMPULSI**	————————————

Essendo realizzato conformemente al modello campione testato con esito positivo ai fini EMC e LVD (rapporto 4446ce3.doc del 13 Luglio 2007), 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:
- 2014/35/UE del 26 Febbraio 2014;
- L. 791 del 18 Ottobre 1977 e s.m.

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

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

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

Fabrizio 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.

Progettazione e produzione di sistemi di pagamento, accessori per videogames e macchine vending
Design and manufacture of payment systems, accessories for videogames and vending machines

Via Ca' Bianca 421
40024 Castel San Pietro
Terme (BO) – ITALY

Tel. + 39 051 944 300
Fax. + 39 051 944 594

http://www.alberici.net

info@alberici.net