HopperOne S11 ccTalk Italy CRYPTO AeS 256

Manuale d'uso

Rev. 1.02

HOPPER ONE S11
ccTalk CRYPTO DH AeS 256
Standard / Reverse

Manuale d'uso

CE

Alberici

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

NOTA

Ogni possibile cura è stata posta nella redazione del presente manuale. Ciò nonostante, non è possibile garantire in ogni momento la corrispondenza assoluta delle descrizioni, in esso contenute, con le caratteristiche del prodotto.
La Alberici S.p.A. declina ogni e qualsivolglia responsabilità verso l'utilizzatore con riferimento a danni, perdite, o reclami di terze parti, conseguenti all'uso del prodotto o causate da errate interpretazioni del presente manuale.
Alberici S.p.A. si riserva il diritto di modificare, senza preavviso e in qualunque modo, qualsiasi parte del presente manuale.

Sommario

STORICO REVISIONI			
Revisione n°	**Data**	**Modifica**	**Note**
Creazione 1.00	21.09.05	Creazione HopperOne	
Rev. 1.01	16.04.16	Modifica testata e nome S11	
Rev. 1.02	30.01.17	Sensori livello ottici	

1. Descrizione generale

1.1 Introduzione

Congratulazioni per l'acquisto dell'erogatore di monete Alberici HopperOne S11! Questo erogatore, progettato e realizzato nei laboratori di ricerca Alberici ricorrendo alle più moderne tecnologie, nasce per soddisfare le esigenze del mercato dei pagamenti automatici.
La tecnologia implementata permette all'HopperOne S11 di contare autonomamente le monete da erogare, e di fermarsi nel caso sia vuoto.
Si integra con facilità sia in giochi con vincita che in apparecchi per il cambio in monete o per l'acquistoi di gettoni.
Queste caratteristiche lo rendono facilmente compatibile con tutte le schede normalmente disponibili sul mercato.

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.

Il montaggio e lo smontaggio dell'hopper dalla sua base a slitta deve avvenire ad alimentazione spenta.
L'installazione deve essere eseguita come da specifiche al paragrafo 2.3 e nella sezione 3.

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.

2. Caratteristiche

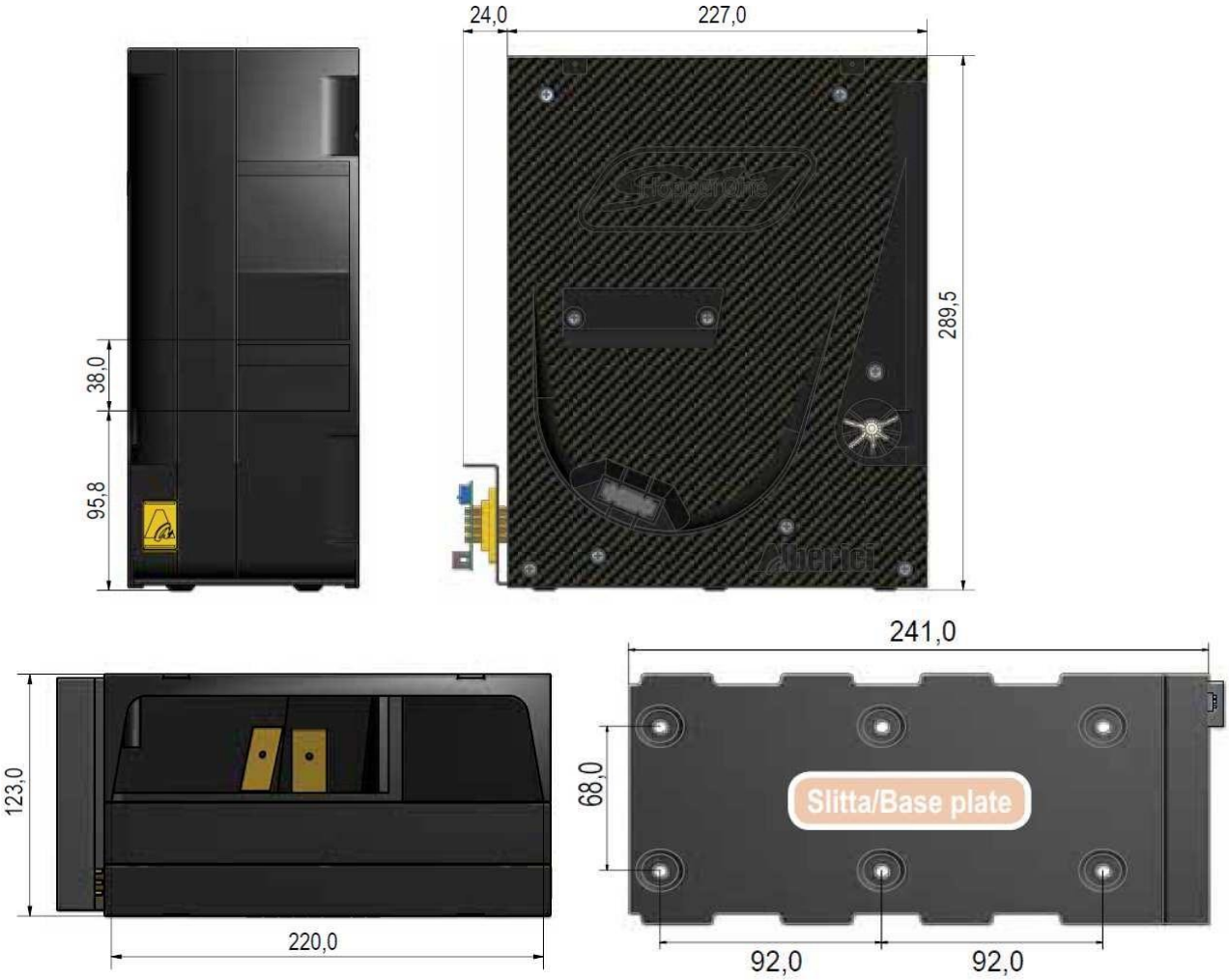
L'HopperOne S11 ccTalk è disponibile in 2 distinte versioni, che si differenziano a seconda delle posizioni rispettive fra il connettore Chinc e l'uscita monete. Quando sono situate sui lati opposti, la versione è detta "STANDARD"; quando si trovano sullo stesso lato, la versione è detta "REVERSE".

Le caratteristiche dell'HopperOne S11 lo rendono intercambiabile con gli altri hopper cosiddetti "universali" presenti sul mercato, oltre che con il suo naturale predecessore HopperOne, di cui ha identiche risposte agli stessi comandi ccTalk.
Può gestire qualsiasi moneta il cui diametro sia compreso tra 16mm e 32mm (a scelta fra 2 catene: una da 16-24mm, e una da 22-32mm), e il cui spessore sia fra 2,0 mm e 3,4 mm.

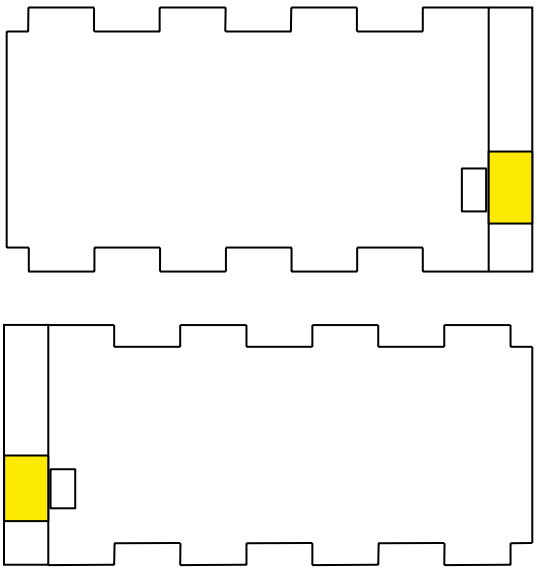
SPECIFICHE TECNICHE TECHNICAL SPECIFICATION

Peso Weight	2 Kg
Alimentazione Operating voltage	24 Vdc
Assorbimento Current draw	1,2A
Temperatura di lavoro Operating temperature	0°C ÷ 50°C
Umidità Humidity	20% ÷ 75%
Protocollo Interface	ccTalk - Aes 256 D.H. ÷ Aes 1024 D.H. (intercon-vertibili interchangable) - Pulse/ccTalk modif.
Capienza Capacity	1.200 monete coins
Velocità Speed	240 monete/min coins/min
Diametro monete Coins diameter	21 ÷ 32 mm, o or 16 ÷ 24 mm
Spessore monete Coins thickness	1,7 ÷ 4,1 mm

2.1 Dimensioni *(inclusa base a slitta)*



2.2 Posizione dei Connettori



Versione Reverse

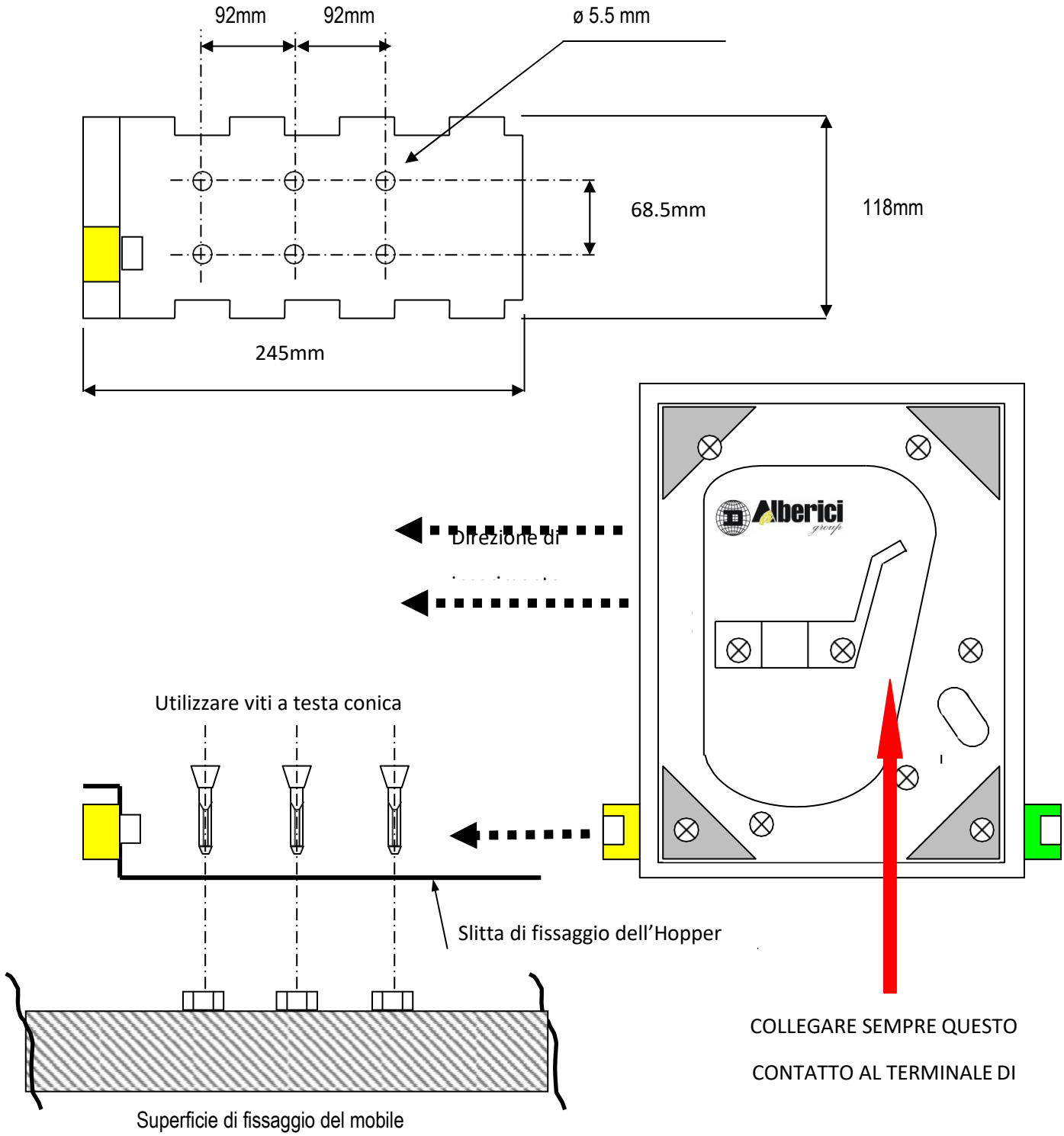
(connettore sul lato di uscita delle monete)

Versione Standard

(connettore sul lato opposto a quello di uscita delle monete)

2.3 Installazione

Per montare l'apparecchio:
. fissare sul mobile la piastra di base: su questa andrà agganciato l'hopper
. inserire l'hopper, facendolo slittare sulla piastra di base fino a completa battuta
. effettuare i collegamenti elettrici (vedi capitolo 3)

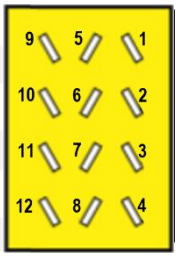


3. Caratteristiche elettriche

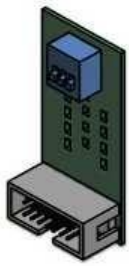
Tutti i segnali gestiti dall'hopper sono in **logica negativa**: il segnale è **attivo** quando si trova **a livello BASSO (GND)**.

3.1 Pinout connettore Cinch e connettore schedino 10poli

12 PIN CCTALK cinch			
GND	1	2	VOID
VOID	3	4	Address 1
DATA cctalk	5	6	VOID
VOID	7	8	Address 2
+24V	9	10	VOID
VOID	11	12	Address 3



10 PIN CCTALK			
Data ccTalk	1	2	VOID
VOID	3	4	GND
VOID	5	6	VOID
+24V	7	8	GND
VOID	9	10	+24V



Vista frontale della spina del cavo

Versione Standard: connettore opposto all'uscita monete; **Versione Reverse**: stesso lato dell'uscita monete.

⚠ La scheda del modello con sensori di livello metallici è diversa dalla scheda del modello con sensori ottici. Se i controlli di livello sono a sensori ottici, collegare gli elettrodi metallici dell'hopper alla terra della macchina.

3.2 Alimentazione

L'HopperOne S11 deve essere alimentato con +24V in tensione continua sul **pin 9 (pin 1 = GND)**, se si usa il connettore CINCH. Sul connettore 10p, i +24V vanno forniti sul pin 7; collegare la terra al pin 8. La velocità di erogazione delle monete è proporzionale alla tensione di alimentazione del motore. Si raccomanda di non scendere mai al di sotto di 18Vdc e di non superare i 27Vdc.
Con una tensione di 24 Vdc la **quantità massima** di gettoni erogabili è di 240 al minuto; aumentare la tensione di alimentazione non fa aumentare la velocità di erogazione.

COLLEGARE SEMPRE IL TERMINALE DELL'HOPPER QUI INDICATO AL TERMINALE DI TERRA DELLA MACCHINA, PER PREVENIRE EVENTUALI DANNEGGIAMENTI DOVUTI ALLE CARICHE ELETTROSTATICHE INTRODOTTE CON LE MONETE.

SETTAGGIO HARDWARE dell' INDIRIZZO SERIALE

Attenzione: l'hopper legge la configurazione dei dip-switch solo al reset.

Add. Sel 1 Switch n° 1	Add. Sel 2 Switch n° 2	Add. Sel 3 Switch n° 3	Indirizzo Seriale *Serial Address*
			3
		ON	4
	ON		5
	ON	ON	6
ON			7
ON		ON	8
ON	ON		9
ON	ON	ON	10

Consumi:

		A riposo	A vuoto	Sotto carico	In blocco (*)
Scheda	(+24Vdc)	20mA/0,24W	20mA/0,24W	40mA/0,48W	40mA/0,48W
Motore	(+24Vdc)	0mA/0m W	70mA/1,4W	1,2 A/28.8W	1,5mA*/30W
Totale		20mA/0,24 W	90mA/1,64W	1,24A/29,28W	1,54 /30,48W

* Il consumo del motore in blocco è limitato elettronicamente, e l'assorbimento qui indicato sarà reale solo per pochi secondi.

4. Comandi Protocollo ccTalk e AES 256 (criptato)

4.1 COMANDI PROTOCOLLO CCTALK NEWSLOT

I comandi ccTalk implementati in questa periferica sono quelli presenti sul documento "Elenco comandi del protocollo ccTalk per il mercato Italiano", legge 289 – comma 6" che specifica il pacchetto di comandi ccTalk Italy attualmente in uso (cfr. più sotto "ccTalk Italy communication protocol"), ma modificati in modo da rendere la periferica conforme ai nuovi requisiti di sicurezza stabiliti con il documento "Technical Table Report 2012, 3.3, 2nd edition - Peripherical 27.02.2013", allegato, a cui si rimanda per i dettagli.

COMANDI DA IMPLEMENTARE per L'OMOLOGAZIONE delle SCHEDE GIOCO "COMMA 6A +" (normativa NEWSLOT)

Per la convenienza del programmatore, si riporta qui sotto lo schema riassuntivo dei soli comandi non-cifrati. **Compaiono in grassetto i comandi specifici del protocollo di cifratura Diffie-Hellmann AeS256, previsto dalla normativa New Slot.**

Tutte le istruzioni sono in lingua Inglese, come dal documento "Technical table report".

NOTA: per informazioni specifiche sul protocollo Diffie Hellmann come definito dal tavolo Tecnico 3.3, vedere il documento allegato "Technical table rev 3 3-2nd edition – Peripheric.pdf".

Code	Command header

HEX	HEADER	DESCRIPTION
EB	***235***	***Read DH public key***
EA	***234***	***Send DH public key***
CE	***206***	***Switch encryption AES key***
C8	***200***	***Request product parameters***

NEW COMMANDS SPECIFICATION

Header 235 - Read DH public key

Not encrypted command

The command sent by the host:

[Destination Address] [1] [Source Address] [235] [Mode] [CHK]

[mode]

0 - request status

1 - request public key

The answer sent by slave when [Mode]= 0 can be:

[Destination Address] [Number Byte] [Source Address] [0] [Status] [CHK]

Where: [status]

0 - shared key calculation in progress

1 - shared key ready

The answer sent by slave when [Mode]= 1 can be:

[Destination Address] [Number Byte] [Source Address] [0] [Status] [CHK]

Where: [status]

0 - shared key calculation in progress, Or

[Destination Address] [Number Byte] [Source Address] [0] [key 1] [key 2]… [key N] [CHK]

Where:

[Number Byte]= N/16 where N is: 32 or 64 or 128 or 256 or 512

[key]

The key length depends on the maximum prime size supported by the hardware.

If mode = 1 then the host can request the DH public key from the peripheral. This is returned LSB first. The length will depend on the processing capability of the peripheral e.g. 128 bytes for 1024 bit DH; it may be much shorter though.

Once the host sends the DH host public key using the command below, it can poll the peripheral status to see when it has performed all the calculations necessary to create the shared key for AES-256 encryption. The host sends [mode] = 0 and the peripheral returns a status of 1 when it is ready.

The host should not attempt any other command until the shared key is ready.

Header 234 - Send DH public key
Not encrypted command
The command sent by the host:
[Destination Address] [Number Byte] [Source Address] [234] [key 1] [key 2]… [key N] [CHK]
Where:
[Number Byte]= N/16 where N is: 32 or 64 or 128 or 256 or 512
[key] is sent LSB first
Received data : ACK
This command transfers the DH host public key to the peripheral. **The host, for an optimized security level, should transfer the key with the maximum length supported either the host and the slave (use the command 200 to handshake the DH key length), but also a DH key length smaller is acceptable.** An ACK is returned immediately and the calculation of a shared key in the peripheral is then started. The host can use the previous command in a status polling mode to find out when the calculation is complete.

Header 223 - Modify inhibit and override registers ***(valid only for coin acceptor - comando valido solo per gettoniera)***
New encrypted command
[Destination Address] [32] [Source Address] [220] [AES data 1]……[AES data 32] [CHK]
The format of Data Blocks [AES data 1].....[AES data 32] when decrypted has this format:
[Number Data Bytes= 13] [223] [Challenge LSB] [Challenge MSB]
[Inhibit mask 1] [Inhibit mask 2] [Inhibit mask 3] [Inhibit mask 4]
[Cash value 1] [Cash value 2] [Cash value 3] [Cash value 4]
[Coin count 1] [Coin count 2] [Coin count 3] [Coin count 4]
[Sorter override mask]
[Filler 1= 0]…[Filler 13= 0] [CRC LSB] [CRC MSB]
Received data : ACK
This command controls coin acceptance in the coin acceptor; both whether a coin is accepted or rejected, and where the coin is routed (e.g. hopper or cashbox).
The inhibit mask controls which coins are enabled or disabled. The cash value sets a maximum value that can be accepted (coins are disabled by the peripheral accordingly). The coin count sets a maximum number of coins that can be accepted (coins are disabled by the peripheral accordingly). The sorter override mask forces subsequent hopper coins to cashbox.
[inhibit mask]
B0 - Coin Type 1 (0 - coin inhibited, 1 - coin enabled)
B1 - Coin Type 2
B2 - Coin Type 3
…
B31 - Coin Type 32
Each bit controls a coin inhibit. Inhibit mask 1 is the LSB and controls coin types 1 to 8; inhibit mask 4 controls coin types 25 to 32.
There is support for up to 32 coin types. Refer to the product manual to see how many coin types are supported.
The default operation is to have all inhibit mask bytes set to 255. There is no problem enabling unsupported coin types.
[cash value]
0 - disabled
1 or more - maximum cash value which can be accepted
This is a single value in 4 bytes (0 to 4,294,967,295). Cash value 1 is the LSB.
The cash value is specified in terms of the lowest monetary unit in that currency e.g. cents, pence, pesos etc.
The coin acceptor will accept coins up to the value specified (and using the inhibit mask supplied). Inserted coins which take the cumulative total above this amount will be rejected regardless of the inhibit status. Event code 1, 'Reject coin', will be returned for these coins. Whenever the host sends this command the cumulative total is reset to zero.
The use of this field alleviates the problems with latency while waiting for a serial credit and trying to apply an inhibit mask before the next coin is entered. Latency is usually worse when high-strength encryption is being used.
[coin count]
0 - disabled
1 or more - maximum number of coins which can be accepted
This is a single value in 4 bytes (0 to 4,294,967,295). Coin count 1 is the LSB.

The coin acceptor will accept coins up to the total number specified (and using the inhibit mask supplied) and then reject all subsequent coins. Event code 1, 'Reject coin', will be returned for these coins. Every time the host sends this command the cumulative coin count is reset to zero.
The use of this field alleviates the problems with latency while waiting for a serial credit and trying to apply an inhibit mask before the next coin is entered. Latency is usually worse when high-strength encryption is being used.
[sorter override mask]

B0 - Sorter Path 1 (0 - cashbox sorting, 1 - normal sorting)	B4 - Sorter Path 5
B1 - Sorter Path 2	B5 - Sorter Path 6
B2 - Sorter Path 3	B6 - Sorter Path 7
B3 - Sorter Path 4	B7 - Sorter Path 8

Each bit controls a sorter path divert; not a coin type. Up to 8 sorter paths (= hoppers) are supported. If a sorter path has an override then the coin will be sent to an auxiliary or default path; usually the cashbox.
The default operation is to have the sorter override mask set to 255.

Header 206 - Switch encryption AES key
New encrypted command: [Destination Address] [48] [Source Address] [220] [AES data 1]……[AES data 48] [CHK]
The format of Data Blocks [AES data 1].....[AES data 32] when decrypted has this format:
[Number Data Bytes= 32] [206] [Challenge LSB= 0] [Challenge MSB= 0]
[New key 1] …[New key 32]
[Filler 1= 0]…[Filler 10= 0] [CRC LSB] [CRC MSB]
Received data : ACK
Only when the device decrypts the packet and confirms the CRC is correct does it change the key to the new value.

Header 200 - Request product parameters
Not encrypted command
The command sent by the host: [Destination Address] [1] [Source Address] [200] [Maximum DH key length] [CHK]
The answer sent by slave is: [Destination Address] [42] [Source Address] [0] [Maximum DH key length] [Maximum Baud rate supported] [FW rev 1] ……[FW rev 8] [ACMI protocol rev 1] [ACMI protocol rev 2] [Serial number 1]……[Serial number 4] [Manuf 1]…[Manuf 16] [Product code 1]…[Product code 8] [DH counter 1] [DH counter 2] [CHK], where:
[Maximum DH key length] follows the table below

DH key lenght	*Maximum DH key length value*
256	2
512	3
1024	4
2048	5
4096	6

[Maximum Baud rate supported] follows the table below:

Baud rate	*Maximum Baud rate value*
9600	1
14400	2
19200	3
38400	4

[FW rev] is 8 x ASCII characters
[ACMI protocol] is a 2 bytes encoding of the release number of this document (e.g. referring the document 3.0 the [ACMI protocol rev 1]= 3 and [ACMI protocol rev 2]= 0)
[Serial Number] is 4 bytes binary. Range 0-4294967296. Serial number 1 = LSB, Serial number 4 = MSB.
The serial number is numerical only and must be unique for a given manufacturer / product name.
[Manuf] is 16 x ASCII characters. Manufacturer name. Upper case only.
[Product code] is 8 x ASCII characters. This is the product name. It may be abbreviated to fit within the 8 characters. Left justified and right padded with spaces (ASCII 32). May contain upper / lower case.
[DH counter] is the counter of the total DH key exchange procedure completed with success in the whole lifecycle of the device. Range 0-65535. DH counter 1 = LSB, DH counter 2 = MSB.

4.2 COMANDI PROTOCOLLO CCTALK ITALY (*ccTalk Italy communication protocol)*

cctalk® communication protocol is the Money Controls serial communication protocol for low speed control networks. It was designed to allow the interconnection of various cash handling devices (*Hoppers, Bill validators, Coin selectors etc.*), mostly in AWP and gaming Industry, but allso 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 Hopper 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 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.

1.3 Connection
The connection of Hopper 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.

1.4 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 exeption of message structure when device answer to instruction Address poll and Address clash. 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 called “broadcast” address and address 1 is default host address. Recommended address values of different devices are shown in Table 1 below.

Device category	Address	Additional addr.	Note
Coin Acceptor	2	11 - 17	Coin validator, coin selector, coinmech...
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 Hopper is factory set as 3; the user can change the default address by using the MDCES commands Address change or Address random or by setting Hopper dip-switches. 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 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 (252 bytes of data, source address, header and checksum: total of 255 bytes).

1.4.3 Command headers (*Instructions*)

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

Header 0 stands for **ACK** (***ack****nowledge*) replay of device to host.

Header 5 stands for **NAK** (***N****o* ***ac****knowledge*) 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 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 mechs
- MDCES commands

Details for use of all instruction 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. 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 (See Error handling).

For noisy environment or higher security application it is possible to use more complex, 16 bit CRC CCITT checksum based on a polynomial of:

x16 + x12 + x5 + 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.

1.5 Timing specification

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

1.5.1 Time between two bytes

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

1.5.2 Time between command and replay

The time between command and reply is dependent on application specific for each command. Some commands return data immediately, and maximum time delay should be within **10 ms**. Other commands that must activate some actions in device may return reply after the 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 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. Hopper Command header set

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

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 above.
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 their 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 it is not connected, or not powered, or simply 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.
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 (usually label printed on electronic circuit board).

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 byte7 information about address mode and options (for details of description see public document cctalk42-2.pdf).

Address could be stored in different type of memory (*RAM. ROM or EEPROM).* Some devices support address change with MDCES command headers (Address change, Address random). 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..

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 (See next chapter), 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 received

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

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 (decimal 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 dispenses coin from the hopper. Maximum number of coins that the 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 11th 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 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] 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 by which the address of the Hopper can be set.

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

Note that the hopper reads the status of the dip-switches only at reset.

DICHIARAZIONE DI CONFORMITÀ

CE

DIRETTIVA 2014 / 35 / UE - 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 di monete**, identificato univocamente da:

Modello	Configurazione	Tipo	Matricola/Seriale
HOPPERONE S11	☐ **ccTalk 24V** ☐ **Pulse 24V**	☐ **Standard** ☐ **Reverse**	_ _ _ _ _ _ _ _ _ _

Essendo realizzato come il prototipo campione testato con esito positivo ai fini EMC e LVD (rapporto 5968-One S11-ccT.doc) il 06/12/2011, 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: 2013-05 (CEI 61-150);
- CEI EN 60335-2-82: 2005-08 (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 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 – 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.

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