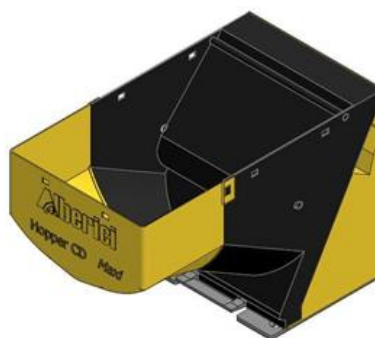


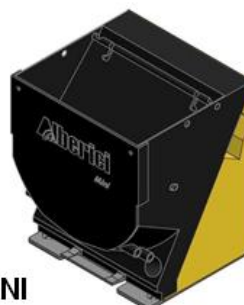
### Hopper CD ccTalk + AES



**MIDI**



**MAXI**



**MINI**



**LATERAL**

Operator's Manual



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](mailto:info@alberici.net)

## NOTICE

This manual has been drafted with the utmost care. Nevertheless, it is not possible to guarantee at all times the absolute correspondence of the descriptions contained therein with the actual characteristics of the product. Alberici S.p.A. declines any and all responsibility towards the User with reference to damages, losses, or claims of third parties, resulting from the use of the product or caused by incorrect interpretations of this manual. Alberici S.p.A. reserves the right to modify, without prior notice and in any way, any part of this manual and the technical specifications of this product, as part of the continuous pursuit of improvement of its products.

# CONTENTS

1.	Safety.....	4
2.	Technical specifications.....	4
3.	Dimensions.....	5
4.	Mounting.....	5
5.	Electrical description.....	6
6.	Commands of the Newslot ccTalk protocol.....	8
7.	ccTalk Italy communication protocol.....	12
8.	Maintenance.....	21
9.	Guarantee.....	22
10.	Customer Service.....	22

STORICO REVISIONI			
Revisione n°	Data	Modifica	Note
Creazione 1.00	04.03.06	Creazione versione italiana	
Rev. 1.01	16.07.14	Nuova versione italiana con aggiornamenti	
Rev. 1.02	26.10.16	Versione con cifratura AES 256	Fw A1.0.3
Rev. 1.03	28.06.19	Nuovi comandi ccTalk FFA2 - FFA3xyy - FFABxx - FFABxxssn	Fw A1.0.6

Congratulations for having purchased our **HopperCD**! This HopperCD has been designed and realized to fulfill all the requirements of the coin-op market. It makes use of the most modern electrical and mechanical technologies, and operates by the safe ccTalk protocol.

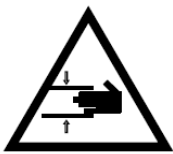
It makes use of a significant quantity of control routines for the management of the internal and external events. It builds-in easily both into **Slot / AWP machines** and into **money changer machines**. Its features make it easily compatible with all the p.c. boards generally available on the market.

## 1. Safety

**The Hopper unit must be installed into systems endowed with ON-OFF mains switch.**

The HopperCD must be connected to and disconnected from its slide connector only when power supply is off. The device includes mechanical parts in motion: DO NOT put your fingers inside it during operation. The installation must be carried out as specified in paragraph 2.3.

Guarantee shall not apply if such instructions are not complied with.



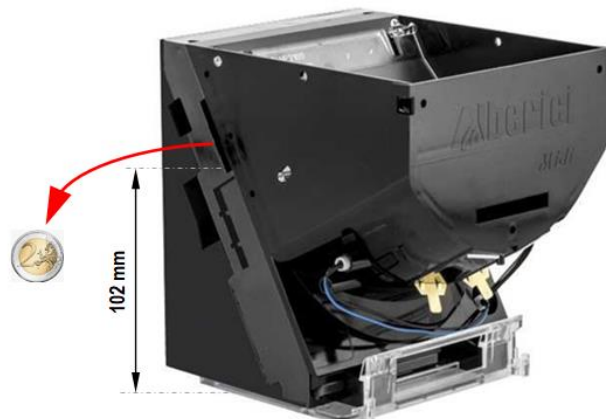
**TAKE CARE:  
HARM DANGER!**  
MECHANICAL PARTS IN MOTION

## 2. Technical Specifications

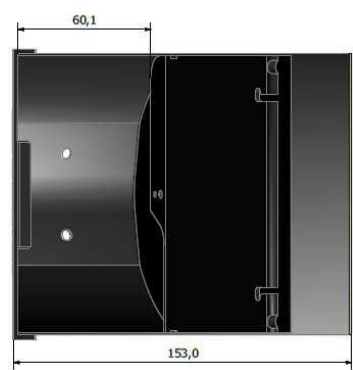
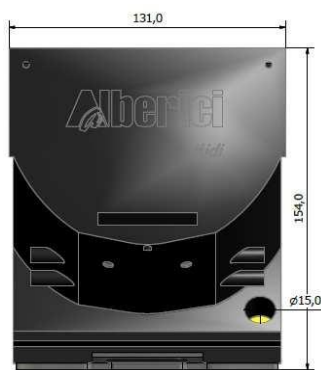
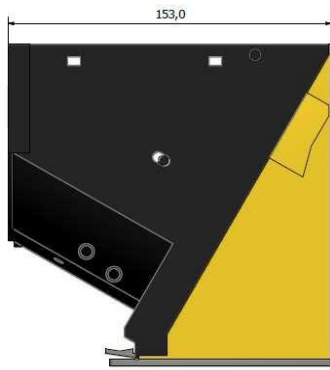
The HopperCD is available in 4 different models, according to capacity and size. The standard features of the HopperCD make it interchangeable with similar devices already existing in the market.

It can handle any coins whose diameter ranges between 20 mm and 26,5 mm. Coin thickness can range between 1,8 mm and 3,5 mm.

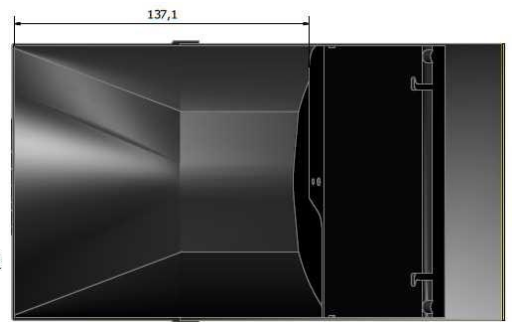
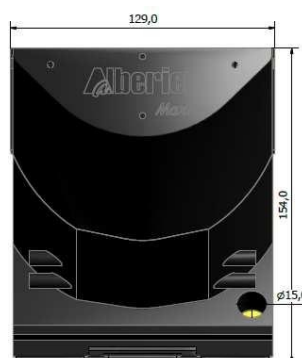
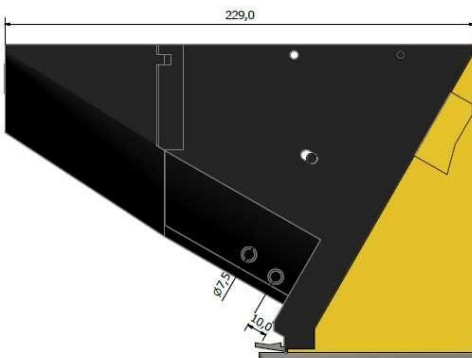
Peso Weight	0,595 Kg (Mini) – 0,660 Kg (Midi) 0,695 Kg (Maxi, Lateral)
Alimentazione Operating voltage	24 Vdc
Assorbimento Current draw	800 mA
Temperatura di lavoro Operating temperature	0°C ÷ 50°C
Umidità Humidity	20% ÷ 75%
Protocollo Interface	Aes 256 D.H. - Aes 2048 D.H. - ccTalk - Pulse
Capienza Capacity	325 (Mini) – 500 (Midi) – 725 (Maxi) – 775 (Lateral)
Velocità Speed	280 monete/min coins/min (7 fori holes) – 240 monete/min coins/min (6 fori holes) 200 monete/min coins/min (5 fori holes)
Diametro monete Coins diameter	18÷ 26,5 mm (disco a 7/5 fori – 7/5 holes disk) 22÷ 31,5 mm (disco a 6 fori – 6 holes disk)
Spessore monete Coins thickness	1,8 ÷ 2,6 mm (dischi 7/6/5 fori – 7/6/5 holes disk)



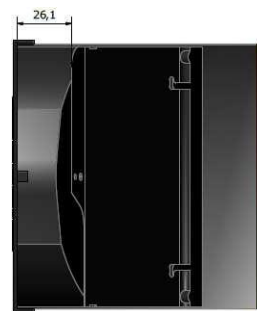
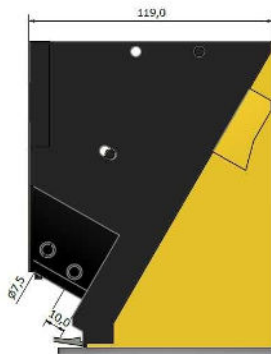
### 3. Dimensions



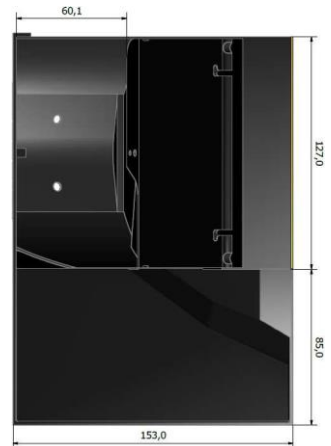
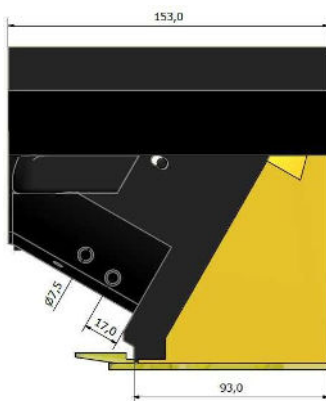
MIDI



MAXI



MINI



LATERAL

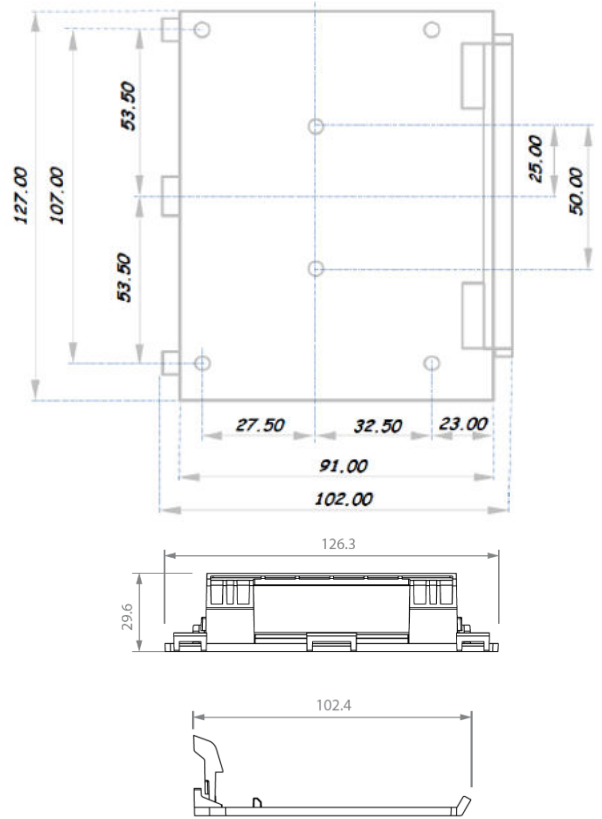
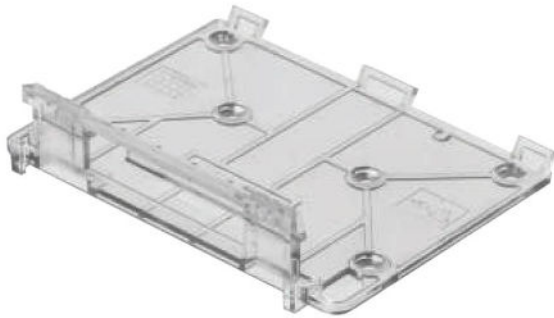
## 4. Mounting

Fasten the polycarbonate base onto the support for the hopper in the cabinet.

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

Before connecting the power supply to the 10p socket located at the rear side of the hopper, please read chapter 5.

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



## 5. Electrical description

### 5.1 Power supply

The voltage supply to the Hopper Discriminator must be 24 V Direct Current (Volts d.c.). The section of wiring must be chosen according to the given current draw specs (see 5.3).

### 5.2 Connector PIN-OUT

The 10-pin ccTalk connector is located behind the Hopper Discriminator, next to the dip-switch row for the serial addressing. All signals are in negative logic, i.e. the signal is active when its potential is equal to GND.

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



### 5.3 Current draw

		Standby		Empty		Loaded (*)	
<b>P.c.board</b>	<b>(+24 Vdc)</b>	<b>40mA</b>	<b>0.48 W</b>	<b>40mA</b>	<b>0.48 W</b>	<b>40mA</b>	<b>0.48 W</b>
<b>Motor</b>	<b>(+24 Vdc)</b>	<b>0mA</b>	<b>0m W</b>	<b>70mA</b>	<b>1.68 W</b>	<b>1 A *</b>	<b>24 W</b>
<b>Total Absorption</b>	<b>(Strength)</b>	<b>0.48 W</b>		<b>2.16 W</b>		<b>24.48 W</b>	

(\*) The current draw of the loaded motor is electronically limited. The shown value is momentarily reached *only in case of blocked device*.

*When the hopper level controls are made through optic sensors, do connect the electrode plates to the machine ground terminals.*

## 5.4 Setting the Hopper Address by Hardware

When needed, i.e. when more than one Hopper Discriminator is used in the same machine, the default serial address of Alberici cctalk Hoppers can be changed via the dip-switches row located on the rear side. It's possible to preset the address of each Hopper combining the three dip-switches in convenient patterns as follows:

(Switch n° 1) (Switch n° 2) (Switch n° 3)			
ON	ON	ON	Indirizzo Seriale
1	2	3	
			3
ON			4
	ON		5
ON	ON		6
		ON	7
ON		ON	8
	ON	ON	9
ON	ON	ON	10

Note that the Hopper reads the status of these lines only at switch on / reset, so dip switch changes during normal operation will have no effect.

## 6. Commands of the Newslot ccTalk protocol

The ccTalk commands implemented in this device are the ones reported on the document "List of the commands of the ccTalk protocol for the Italian market", law 289 - comma 6", which specifies the ccTalk Italy package of commands currently in use (see "ccTalk Italy communication protocol"), but modified so as to make the peripheral compliant with the new security requirements established by the document "Technical Table Report 2012, 3.3, 2nd edition - Peripheric 27.02.2013", to which the reader is referred for details.

### **COMMANDS TO BE IMPLEMENTED ON THE GAME-BOARDS for the "COMMA 6A +" (NEWSLOT) CERTIFICATION**

*For the convenience of the designer, this data sheet provides the sum-up of the specific commands relevant to the Diffie-Hellmann AeS256 encryption protocol, as required by the Italian New Slot regulations. The description of the non-encrypted commands 235, 234, 206 e 200 is given in both italian and english languages.*

**PLEASE NOTICE:** the details concerning the Diffie Hellmann protocol can be found in the attached document "Technical table rev 3 3-2<sup>nd</sup> edition – Peripheric.pdf".

Code		Command header
HEX	HEADER	DESCRIPTION
<i>EB</i>	<i>235</i>	<i>Read DH public key</i>
<i>EA</i>	<i>234</i>	<i>Send DH public key</i>
<i>CE</i>	<i>206</i>	<i>Switch encryption AES key</i>
<i>C8</i>	<i>200</i>	<i>Request product parameters</i>

### **DEFINITION OF THE NON-ENCRYPTED COMMANDS**

#### **HEADER 235 (0xEB) "READ DH PUBLIC KEY"**

*Message structure of the request for the DH Public Key sent by the host:*

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

[mode]

0 - request status

1 - request public key

The slave will reply by one of the two shown below, according to the chosen parameter:

- mode 0:[Destination Address] [Number Byte] [Source Address] [0] [Status] [CHK]

- mode 1:[Destination Address][Number Byte][Source Address][0][key 1][key 2]... [key N][CHK]

*in which:*

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

[status]

0 - shared key calculation in progress

1 - shared key ready

[key]: LSB first

The public key returned by the slave shall be 256bit, i.e. 32 byte transmitted.

#### **HEADER 234 (0xEA) "SEND DH PUBLIC KEY"**

*Structure of the command that transfers the DH host public key to the peripheral.:*

**[Destination Address] [Number Byte] [Source Address] [234] [key 1]**

**[key 2]... [key N ] [CHK]**

*In which:*

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

[key]:LSB first

The host transfers its 256bit DH public key to the peripheral, i.e. 32 bytes transmitted.

Upon receiving the key, the slave shall return ACK and will start the calculation of a shared key.



## HEADER 200 (0xC8) "REQUEST PRODUCT PARAMETERS"

By this command it is possible to request the parameters that allow identification of the peripheral unit.

The host sends its maximum DH key length as command parameter: **[Destination Address] [1] [Source Address] [200] [Maximum DH key length] [CHK]**

The slave returns:

**[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]**

- [Maximum DH key length]: 2 (256 bit);
- [Maximum Baud rate supported]: 1 (9600);
- [FW rev 1] ... [FW rev 8]: "4.02 ";
- [ACMI protocol rev 1] [ACMI protocol rev 2]: 33;
- [Serial number 1] ... [Serial number 4]: 0-4294967296, LSB->MSB;
- [Manuf 1] ... [Manuf 16]: "ALBERICI";
- [Product code 1] ... [Product code 8]: "AH MNMCR";
- [DH counter 1] [DH counter 2]: 0-65535, LSB->MSB;

## STRUCTURE OF THE ENCRYPTED MESSAGES

Structure of the messages - with data and without data:

[ Destination address ]

[ Nr. of AES bytes ]

[ Source address ]

[ Header ]

[ AES 1 ]

...

[ AES n ]

[ Checksum ]

The number of AES bytes is multiple of 16 (16, 32, 64, 256, ...). The header is a constant equal to '220' and is a constant value to recognize the encrypted command: the slave reply shall be '0'.

The CHK is the simple 8 bit checksum

The format of AES Data Blocks for a message with data - before encryption - has the following structure:

**[ nr. bytes ] [ header ] [ challenge lsb ] [ challenge msb ] [ data 1 ] [ data 2 ] ... [ data n ] [ filler 1 ] [ filler 2 ] ... [ filler k ] [ crc lsb ] [ crc msb ]**

nr. bytes = number of data bytes n

header = ccTalk command

challenge = number sent by the host

data = ccTalk data

filler = 0 for byte padding (*ignore*)

crc = error-detecting code (CRC-CCITT, polynomial  $x^{16}+x^{12}+x^5+1$ , initial 0x0000)

The format of AES Data Blocks for a message without data - before encryption - has the following structure:

**[ nr. bytes ] [ header ] [ challenge lsb ] [ challenge msb ] [ filler 1 ] [ filler 2 ] ... [ filler k ] [ crc lsb ] [ crc msb ]**

1. nr. bytes = 0

2. header = ccTalk command or "0=ACK", "5=NACK", "6=BUSY"

3. challenge = number sent by the host

4. filler = 0 for byte padding (*ignore*)

5. crc = error-detecting code (CRC-CCITT, polynomial  $x^{16}+x^{12}+x^5+1$ , initial 0x0000)

## NEW COMMANDS SPECIFICATION

(from "Technical Table Report 2012, 3.3, 2nd edition - Peripheric 27.02.2013")

### Header 235 - Read DH public key (see page 12 as well)

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

[DestinationAddress][NumberByte][SourceAddress][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 (see page 12 as well)

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)

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. Default operation is to have all inhibit mask bytes set to 255. There is no problem enabling unsupported coin types.

[cash value]

0 - disabled

0 or more - maximum cash value that 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, 'Rejectcoin', will be returned for these coins. Every time 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 ) B1

- Sorter Path 2

B2 - Sorter Path 3

B3 - Sorter Path 4

B4 - Sorter Path 5

B5 - Sorter Path 6

B6 - Sorter Path 7

B7 - Sorter Path 8

Each bit controls a sorter path divert; not a coin type. Up to 8 sorter paths ( = hopper ) 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** (see page 13 as well)

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:

<i>DH key lenght</i>	<i>Maximum DH key length value</i>
256	2
512	3
1024	4
2048	5
4096	6

[Maximum Baud rate supported] follows the table below:

<i>Baud rate</i>	<i>Maximum Baud rate value</i>
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.

**PLEASE NOTICE: for specific and more detailed information about the Diffie-Hellman protocol, as defined by the Technical Table 3.3, please see the "Technical table rev 3 3-2nd edition - Peripheric.pdf" papers.**

## 7. 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 also in other devices that use those components.

**ccTalk®** is an open standard.

All documentation is available at web site: [www.cctalk.org](http://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 derived from RS232 standard.

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

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

RS232 handshaking signals (*RTS, CTS, DTR, DCD, DSR*) are not supported.

Message integrity is controlled by means of checksum calculation.

#### 1.1 Baud rate

The baud rate of 9600 was chosen as compromise between cost and speed.

Timing tolerances is same as in RS232 protocol and it should be less than 4%.

#### 1.2 Voltage level

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

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

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

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 exception 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** (*acknowledge*) replay of device to host.

**Header 5** stands for **NAK** (*No acknowledge*) replay of device to host.

**Header 6** is **BUSY** replay of device to host.

In all three cases no data bytes are transferred. Use of ACK and NAK headers 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 (*Binary Coded Decimal*) numbers, Hex numbers or ASCII strings. Interpretation as well as format is specific to each header use, and will be explained in separate chapter.

### 1.4.5 Checksum

Message integrity during transfer is checked by use of simple zero checksum calculation.

Simple checksum is made by 8 bit addition (modulus 256) of all the bytes in the message. If message is received and the addition of all bytes are non-zero then an error has occurred (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:

$x^{16} + x^{12} + x^5 + 1$  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.

## 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 byte 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] [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] [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 dispense 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] Broadcast 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] Broadcast message

Hopper answer: [01] [00] [00] [00] [FD] ACK Address is changed

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

## 8. Maintenance

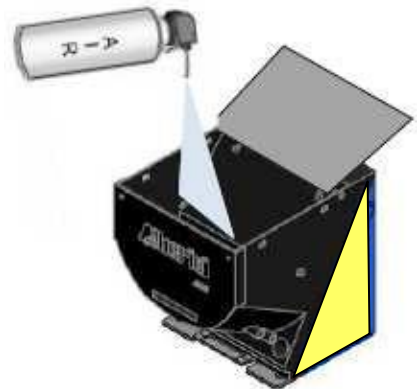
Prior to any maintenance operation, switch power off and disconnect the power cord. Clean the disk of the Hopper Discriminator every 100.000 pay-outs, by pressurized air.

The sloped shape of the optic sensor housing prevents the build-up of the dirt, so reducing the need to clean it too frequently. Whenever you clean the disc, it is anyway advised to clean also the output optic sensor.

Check from time to time that the reservoir and the disc do not contain debris or bent / warped coins, and always remove them, since they can obstruct the output or the rotation, hamper the operation of the hopper, damage the Hopper parts and thwart its performance.

To clean the Hopper, pull up its flap, and blow dry pressurized air onto the disc parts and the sensors slots (accessible through the coin pits), as well as through the coin outlet slot.

CAUTION: if any parts need to be dismantled for deeper cleansing or other maintenance operations, please send the Hopper Discriminator to Alberici S.p.A., that will properly carry out the necessary service repairs, cleaning and updating.



## 9. Guarantee

The manufacturer will fix malfunctions arising from production faults in this machine or parts of it within 12 months from the date of sale.

All communications referring to guarantee repairs or replacements must be accompanied by the product serial number and the copy of the sale invoice.

To obtain your guarantee repair, please send the part to the Dealer where you purchased the machine, together with the following documents:

- copy of the sale invoice
- delivery note stating "returned for guarantee repair"
- detailed report of the problem found and the circumstances in which it occurs.

Before sending the product, please get in touch with your Dealer or with Alberici S.p.a. (+39 051 944300); very malfunctions can be fixed via a simple phone call, saving you costs and time.

Alberici S.p.a. will verify that warranty is applicable, i.e. that problem is not caused by:

- transport damages
- damages from incorrect installation or wrong configuration
- installation in premises or areas not complying with the prescribed safety requirements
- intentional or unwilling tampering
- wrong or careless use or maintenance
- non-compliance with precautions prescribed (see Chapter 4. Caution)
- natural disasters, vandalisms, intentional or unintentional damage

Guarantee is considered automatically expired if outer and inner labels are missing.

Transport costs of repaired products are at the Customer's charge.

## 10. Customer Service

Alberici S.p.a. will be pleased to offer all the necessary information on use, ordinary maintenance and technical service. Please call (+39) 051 944300 and specify if your request concerns information on use or technical support.



## DICHIARAZIONE DI CONFORMITÀ



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

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

### DICHIARA

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

Modello	Configurazione	Tipo comunicazione	N° di Serie e/o matricola
<b>HOPPER CD</b>	<input type="checkbox"/> Mini <input type="checkbox"/> Midi <input type="checkbox"/> Maxi <input type="checkbox"/> Lateral	<input type="checkbox"/> CCTALK <input type="checkbox"/> IMPULSI <input type="checkbox"/> AES	-----

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 li, \_\_\_/\_\_\_/\_\_\_\_\_

Il Presidente

### Alberici S.P.A.

Progettazione e produzione sistemi di pagamento, accessori per videogames e vending machines

Via Ca' Bianca 421, 40024 Castel San Pietro Terme (BO), Italia

Telefono: +39-(0)51-944300 r.a. – Fax: +39-(0)51-944594 – P.Iva: 00627531205

E-mail: [info@alberici.net](mailto:info@alberici.net) – Url: <http://www.alberici.net>







## NOTICE

Alberici S.p.A. reserves the right to make changes to the equipment described and to its technical specifications at any time and without notice, in pursuit of continual improvement of this product.



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](mailto:info@alberici.net)