

HopperCD Evolution II

User Manual

Rev. 1.04

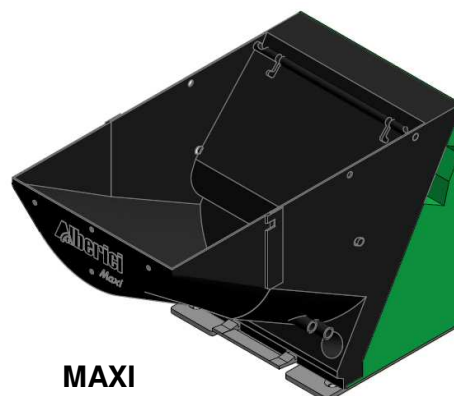
HopperCD EVOLUTION II

Mini - Midi - Maxi - Lateral

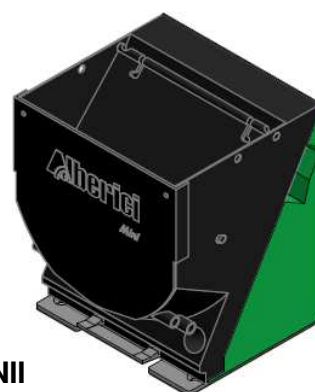
24Vdc ccTalk



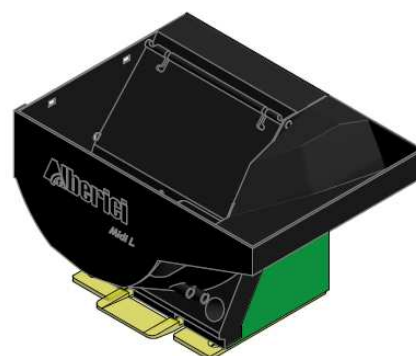
MIDI



MAXI



MINI



LATERAL

User manual



Alberici[®]
CASH SOLUTIONS

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

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

Tel.: +39.051.944300
Fax.: +39.051.944594

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

1. General

Congratulations on your purchase of the Alberici Evolution II Hopper. The Hopper Evolution II has been designed in laboratories and manufactured in the Alberici workshops, to meet the most stringent requirements in payment management.

The technologies implemented make it capable of handling multiple operations, such as to identify different coin values, count them and dispensing them. It can be equipped with a sorter to separate the counted coins - and the rejected ones - in 2, or 3, or 5 different directions: respectively separator CD2 S, CD3 S, and CD5 S.

It is mostly indicated for use in Cross-Changer machines.



1.1 Operation

The Evolution II hopper can recognize and count up to 3 different coin or token denominations (provided their shape is round, without grooves, corners, holes or pronounced reliefs), having diameter 18 mm - 26.5 mm, and 1.8 mm – 2,6 mm thickness.

As it dispenses the coins, it identifies and counts the available denominations, and transmits the information via ccTalk protocol to the host pcb. The recognition of each coin is obtained through an encoder device used to scan 4 different magnetic parameters of the coin, while a special lever with double sensor measures its diameter: the combination of these data allows the accurate identification of the denomination.

The outgoing coin momentarily interrupts the modulated infrared ray between an infra-red emitter and a photo-transistor: each interruption indicates that the delivery of a coin has occurred, the value of which had been previously identified.

The particular encryption of the control signal prevents attempts at fraud by blinding the sensor.

If a temporary block occurs, the electronics of the Hopper Evolution detects the current over-absorption, and temporarily reverses the direction of rotation of the disk, so as to free the output and re-establish the normal flow of supply.

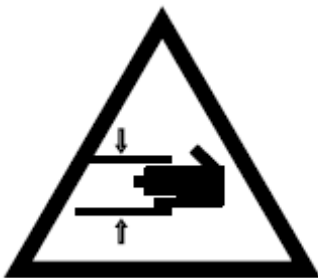
The minimum level of coins in the hopper can be controlled by optical sensors.

The delivery speed is approx. 180 coins per minute in standard conditions.

1.2 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.



CAUTION:
HARM DANGER!

2. Technical Specs

SPECIFICHE TECNICHE TECHNICAL SPECIFICATIONS

Peso Weight	0,595 Kg (Mini) – 0,660 Kg (Midi) – 0,695 Kg (Maxi e 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	ccTalk 24v
Capienza Capacity	325 (Mini) – 500 (Midi) – 725 (Maxi) – 775 (Lateral)
Velocità Speed	210 monete/min coins/min
Diametro monete Coins diameter	18 ÷ 26,5 mm
Spessore monete Coins thickness	1,8 ÷ 2,6 mm

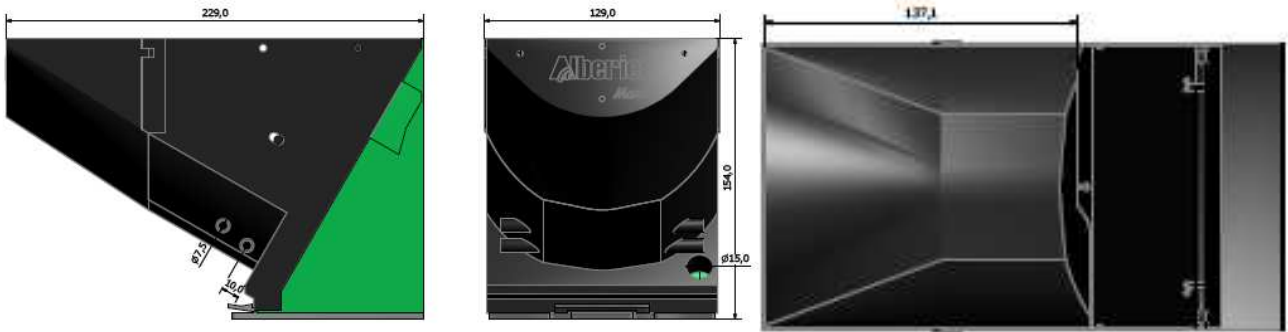
SW release (08.05.2018): v. 6.0.8

3. Mechanical description

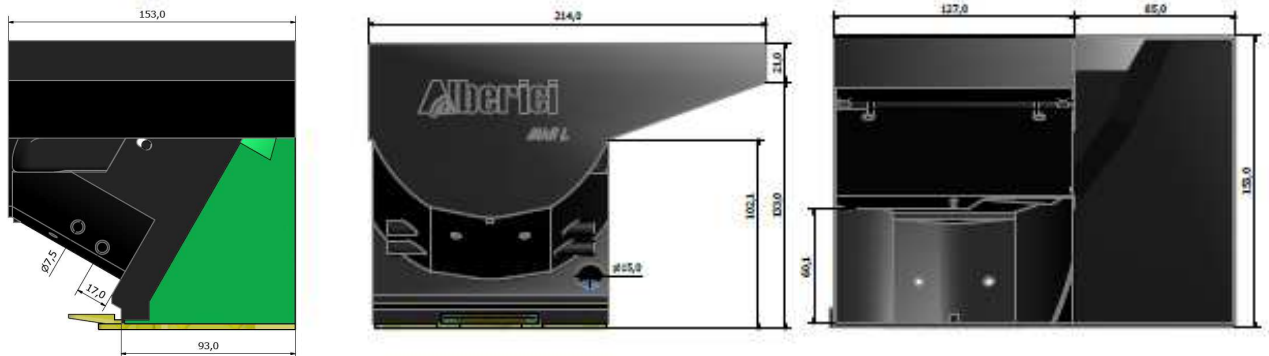
The Hopper Evolution II is available in 4 different models, according to capacity and size. The standard features of the HopperCD make it interchangeable with similar devices already existing in the market. It can handle any coins whose diameter ranges between 20 mm and 26,5 mm. Coin thickness can range between 1,8 mm and 2,6 mm.

3.1 Size





HOPPER EVOLUTION MAXI

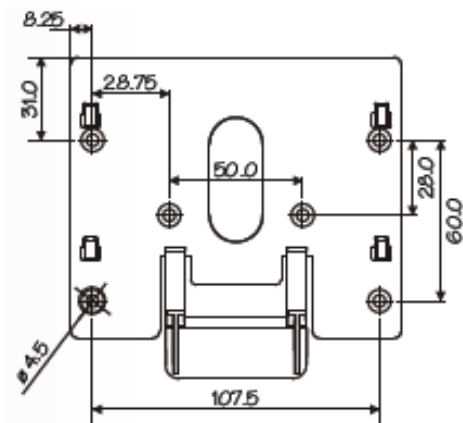
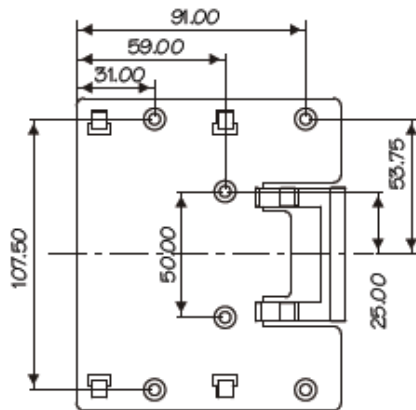


HOPPER EVOLUTION LATERAL

3.2 Installation

Take the following easy steps:

- . fasten the slide polycarbonate support
- . slide the Hopper Discriminator in
- . before plugging in electrical supply, please read chapter 4



REVERSE MOUNT SUPPORT

To remove the hopper, hold the release clip pressed down wise, and slide the hopper off.

4. Electrical description

4.1 Poker Supply

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

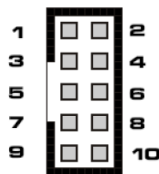
4.2 Current draw

		Standby	Empty	Loaded (*)
PCB	(+24 Vcc)	40mA 0.48 W	40mA 0.48 W	40mA 0.48 W
Motor	(+24 Vcc)	0mA 0m W	70mA 1.68 W	1 A * 24 W
Total		0.48 W	2.16 W	24.48 W

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

4.3 Socket 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 come in negative logics, that is signal is active when voltage is GND.



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

When the hopper level contr machine ground terminals.

ct the electrode plates to the

4.4 Setting the Hopper cctalk address by Dip-Switches

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 serial address of the Hopper by combining the three dip-switches in convenient patterns as follows:

ON	(Switch n° 1)	(Switch n° 2)	(Switch n° 3)	Serial Address
				3
ON				4
	ON			5
ON	ON			6
			ON	7
ON			ON	8
		ON	ON	9
ON	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.

5. Maintenance

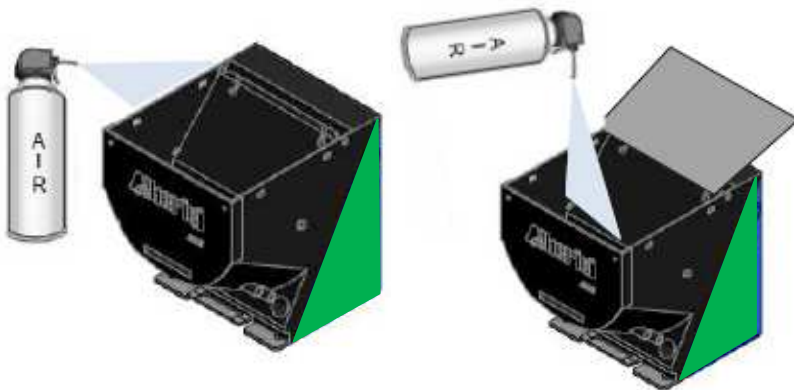
Prior to any maintenance operation, switch power off and disconnect the power cord.
Clean the disk of the Hopper Discriminator every 100.000 pay-outs.

The sloped shape of the optic sensor housing prevents the build-up of the dirt, so reducing the need to clean it too frequently. It is anyway advised to clean regularly 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:

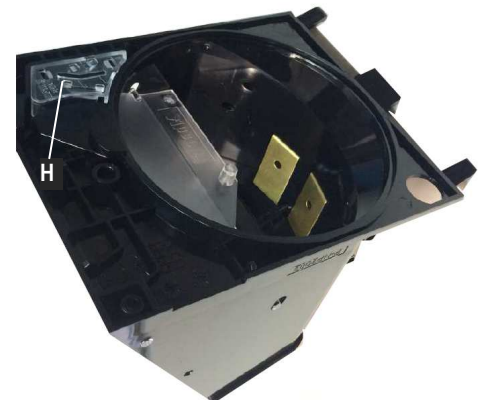
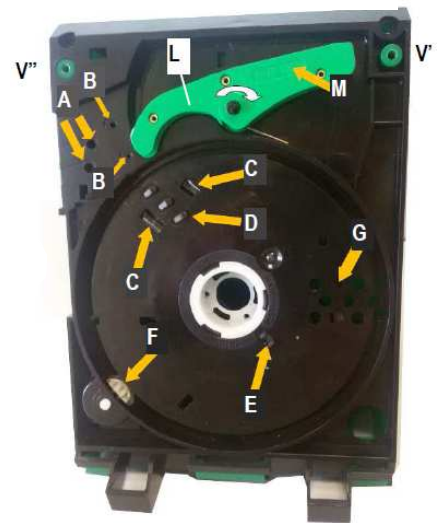
- pull up its flap, and blow dry pressurized air onto the disc parts and the sensors slots (accessible through the coin pits), as well as through the coin outlet slot.



- clean periodically the coin disk, to avoid that the accumulation of dirt can distort correct recognition, so causing erroneous excess or defective calculations.
- do not use alcohol, diluent, turpentine essence, acetone, petrol or other petroleum products, or containing citric acid.
- the sensors lenses are made of transparent polymers, and must be cleaned with great caution so as not to scratch them. Gently repeat several times, until dirt and dusts are completely removed.

Carry out cleaning as follows:

1. If the Evolution II is equipped with CD S sorter, disconnect its cable and disassemble it (it is held by a single fixing screw).
2. Remove screws V' and V'', and extract the hopper from the triangular base, thus accessing the driving disks.
3. Remove the discs and clean the toothed edge of the black one. Remove the impurities from the surfaces of the discs, using a rag moistened with a solution based on Isopropyl Alcohol (do not use denatured alcohol, which could damage the prisms of the sensors).
4. Tilt the coin eject lever L to reveal the sensors located behind it in position M: remove the lever, and blow compressed air into the sensor seats.
5. Clean the output sensors A and B, always blowing compressed air. Clean the prisms C and H.
6. Check the operation of the elastic teeth D, and remove any debris that interfere with their oscillation.
7. Blow compressed air onto the encoder E, and if necessary remove the dirt by a small paint brush or by a mall and soft bottle-brush.
8. Blow compressed air onto the drive gear F, and if necessary remove the dirt by a soft brush.
9. Remove any debris from the G louvres, so as to keep efficient the drainage of the small debris possibly introduced with coins.
10. Blow compressed air on the surface of the plate, and complete the cleaning again by using a rag moistened with a solution based on Isopropyl alcohol (do not use denatured alcohol, which could damage sensor prisms).
11. Put the hopper basket back on its triangular base, and fix it with the two screws V' and V''. Place the sorter in its position at the side of the hopper coin outlet, fasten it by its fixing screw, and connect the power cable.



6. ccTalk protocol

6.1 ccTalk commands

NOTE: the commands necessary to implement the Hopper Evolution II on the Host pcb are listed in the table below.

To view the detailed list of responses by the Evolution II hopper to the commands sent by the host pls. see: § 6.2 ccTalk Protocol (next page).

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

6.2 cctalk Protocol of Hopper Evolution II

ccTalk® communication protocol is the Money Controls (formally Coin Controls) serial communication protocol for low speed control networks. It was designed to allow the interconnection of various cash handling devices (*Hopper, Card reader, Bill validators, Coin selectors etc.*), mostly in AWP and gaming Industry, but also in other devices that use those components.

ccTalk® is an open standard. All documentation is available at web site: www.cctalk.org.

Communication protocol of Alberici Hoppers AH04... is implemented according to generic specification 4.4

1 Communication specifications

ccTalk serial communication is derivation of RS232 standard.

Low data rate NRZ (*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

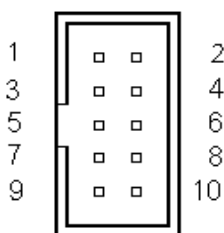
Data I/O line is “open collector” type, so it is possible to use device in systems with different voltage (12V pull up in older devices).

1.3 Connection

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

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

Fig. 1 AH04 - ccTalk connector



PIN Nr.	Function
1	ccTalk Data
2	Not used
3	Not used
4	GND
5	Not used
6	Not used
7	+24 V
8	GND
9	Not used
10	+24 V

Table 1
AH04 - ccTalk
connector

1.4 Message structure

Each communication sequence consists of two message packets.

Message packets for simple checksum case is structured as follows:

[Destination address]

[Nr. of data bytes]

[Source address]

[Header]

[Data 1]

...

[Data n]

[Checksum]

There is an exception of message structure when device answer to instruction 253 "Address poll" and 252 "Address clash". The answer consists of only one byte representing address delayed for time proportional to address value or random delay.

For CRC checksum case format is:

[Destination address]

[Nr. of data bytes]

[CRC 16 LSB]

[Header]

[Data 1]

...

[Data n]

[CRC 16 MSB]

1.4.1 Address

Address range is from address 0 to address 255. Address 0 is special case or so called "broadcast" address and address 1 is default host address. The recommendation for address value of different devices are presented in table 2.

Device category	Address	Additional addr.	Note
Coin Acceptor	2	11 - 17	Coin validator, selector, mech...
Payout	3	4 - 10	Hopper
Bill validator	40	41 - 47	Banknote reader
Card Reader	50		-
Display	60		Alphanumeric LC display
Keypad	70		-
Dongle	80	85	Safety equipment
Meter	90		Replacement for el.mec. counters
Power	100		Power supply

Table 2 Standard address for different types of devices

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

1.4.2 Number of data byte

Number of data byte in each transfer could be from 0 to 252.

Value 0 means that there are no data bytes in the message, and total length of message packet will be 5 bytes. Although theoretically it will be possible to send 255 bytes of data because of some limitations in small micro controllers the number is limited to 252..

1.4.3 Command headers (*Instructions*)

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

Header 0 stands for **ACK** (*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 is explained separately for each specific message transfer.

Commands are divided in to several groups according to application specifics:

- Basic general commands
- Additional general commands
- Commands for Coin acceptors
- Commands for Bill validators
- Commands for Payout
- MDCES commands

1.4.4 Data

There is no restrictions for data format use. Data could be BCD (*Binary Coded Decima*) numbers, Hex numbers or ASCII strings. Interpretation as well as format is specific to each header use, and will be explained in separate chapter.

1.4.5 Checksum

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

Simple checksum is made by 8 bit addition (modulus 256) of all the bytes in the message.

If message is received and the addition of all bytes are non-zero then an error has occurred.

1.5 Timing specification

The timing requirements of ccTalk are not very critical but there are some recommendation.

1.5.1 Time between two bytes

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

1.5.2 Time between command and replay

The time between command and reply is dependent on application specific for each command. Some commands return data immediately, and maximum time delay should be within **10 ms**.

Other commands that must activate actions in device may return reply after action is finished.

1.5.3 Start-up time

After the power-up sequence, Hopper must be ready to accept and reply to a ccTalk message within time period of less than 250 ms. During that period all internal check-up and system settings must be done, and hopper should

1.6 Error handling

If slave device receive the message with bad checksum or missing data no further action is taken and receive buffer will be cleared. Host software should decide to re-transmit message immediately or after a fixed amount of time. In case when host receive message with error it has same options.

2. Hopper Command header set

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

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

Table 3 List of Hopper Evolution ccTalk command headers

Command headers are divided in to 4 different groups:

- Common command headers

- Hopper command headers

- MDCES command headers

2.1 ALBERICI specific command headers

2.1.1 Command header 254 [hexFE], Simple poll

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

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

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

Message format is:

Host sends: [Dir][00][01][FE][Chk]

Hopper answer: [01][00][Dir][00][Chk]

Hopper default address is 3, example of message packet is:

Host sends: [03][00][01][FE][FE]

Hopper answer: [01][00][03][00][FC] ACK message

2.1.2 Command header 246 [hexF6], Request manufacturer ID

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

Message format is:

Host sends: [Dir][00][01][F6][Chk]

Hopper answer: [01][Nr.b][Dir][00][a1][a2]...[an][Chk]

[Nr.b] is number of data bytes-characters sent by Hopper, and a1 to an are ASCII characters. The example of message packet is:

Host sends: [03][00][01][F6][06]

Hopper answer: [01][0E][03][00][41][6C][62][65][72][69][63][69][86]

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

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

Message format is:

Host sends: [Dir][00][01][F5][Chk]

Hopper answer: [01][06][Dir][00][50][61][79][6F][75][74][Chk]

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

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

Host sends: [03][00][01][F5][07]

Hopper answer: [01][06][03][00][50][61][79][6F][75][74][74]

2.1.4 Command header 244 [hexF4], Request product code

Hopper answer with ASCII string of character, representing its factory type.

For Alberici Hoppers it is 'AH EVM 1'. Message format is:

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

Hopper answer: [01][nr.byte][Dir][00][a1][a2]...[an][Chk]

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

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

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

Hopper answer: [01][09][03][00][41][48][20][45][56][4D][20][31][12]

2.1.5 Command header 242 [hexF2], Request serial number

Hopper answer with three byte serial number.

Message format is:

Host sends: [Dir][00][01][F2][Chk]

Hopper answer: [01][03][Dir][00][Ser.1-LSB][Ser.2][Ser.3-MSB][Chk]

The first data byte sent is LSB of serial number.

Example of message packets for Hopper (*address 3*) and serial number 1234567, hex [BC][61][4E] is:

Host sends: [03][00][01][F2][0A]

Hopper answer: [01][03][03][00][4E][61][BC][8E]

2.1.6 Command header 241 [hexF1], Request software revision

Hopper return ASCII string of character representing software version and revision. Message format is:

Host sends: [Dir][00][01][F1][Chk]

Hopper answer: [01][Nr.b][Dir][00][a1][a2]...[an][Chk]

Number of data bytes in ASCII string is not limited and each producer has it's own system of labelling. Example of message packets for Hopper(*address 3*) is:

Host sends: [03][00][01][F1][0B]

Hopper answer: [01][0B][03][00][75][31][2E][30][20][70][31][2E][30][2E][30][70]

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

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

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

2.1.7 Command 197 [hexC5], Calculate ROM checksum

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

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

third is data(in RAM) checksum. Any changes in program memory or data will change the respond of hopper. Message format is:

Host sends: [Dir][00][01][C5][Chk]

Hopper answer: [01][3][Dir][00][Cksum 1][Cksum 2][Cksum 3][Chk]

Example of message string for Hopper(address 3) is:

Host sends: [03][00][01][C5][37]

Hopper answer: [01][03][03][00][53][6E][CC][6C]

2.1.8 Command header 1 [hex01], Reset device

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

Host software must re enable hopper to perform a new payout. Message format is:

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

Hopper answer: [01][00][Dir][00][Chk] ACK message

Example of message packets for hopper (address 3) is:

Host sends: [03][00][01][01][FB]

Hopper answer: [01][00][03][00][FC] ACK message

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

2.2 Hopper specific command headers

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

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

This command allow the reading of High/Low level sensor in payout systems.

Hopper answer with one byte that describe the sensors status.

The meaning of bits in status byte is the following:

Bit 0 - Low level sensor status.

0 – Higher than or equal to low level trigger

1 – Lower than low level trigger

Bit 1 – High level sensor status

0 - Lower than high level trigger

1 - Higher than or equal to high level trigger

Bit 4 - Low level sensor support

0 – Features not supported or fitted

1 - Features supported and fitted

Bit 5 - High level sensor support

0 - Features not supported or fitted

1 - Features supported and fitted

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

Message format is:

Host sends: [Dir][00][01][D9][Chk]

Hopper answer: [01][01][Dir][00][d1][Chk]

Example of message packets for Hopper(address 3) AH24 with no coins(empty) is:

Host sends: [03][00][01][D9][23]

Hopper answer: [01][01][03][00][11][EA]

Only low sensor is supported, and hopper is empty.

2.2.1 Command header 164 [hexA4], Enable Hopper

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

Message string format is:

Host sends: [Dir][01][01][A4][d1][Chk]

Hopper answer: [01][00][Dir][00][Chk] ACK

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

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

Host sends: [03][01][01][A4][A5][B2]

Hopper answer: [01][00][03][00][FC] ACK

2.2.2 Command header 163 [hexA3], Test Hopper

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

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

[Error mask 1]

Bit 0 – Absolute maximum current exceeded

Bit 1 – Payout time-out occurred

Bit 2 – Motor reverse during last payout to clear a jam

Bit 3 – Opto fraud attempt, path blocked during idle

Bit 4 – Opto fraud attempt, short circuit during idle

Bit 5 – Opto blocked permanently during payout

Bit 6 – Power up detected

Bit 7 – Payout disabled

[Error mask 2]

Bit 0 – Opto shorted during payout

Bit 1 – Flash data crc error

Bit 2 – Use other hopper to pay

Bit 3 – NU read 0

Bit 4 – Motor reverse limit end

Bit 5 – Unrecognized coin reverse limit

Bit 6 – Sorter blocked

Bit 7 – PIN mechanism active

Message string format is:

Host sends: [Dir][00][01][A3][Chk]

Hopper answer: [01][02][Dir][00][err 1][err 2][Chk]

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

Host sends: [03][00][01][A3][59]

Hopper answer: [01][02][03][00][C0][00][BA]

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

2.2.3 Command header 134 [hex86], Dispense hopper value

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

Hopper must be enabled before use of this command.

Message format is:

Host sends: [Dir][05][01][86][sn1][sn2][sn3][Val-lo][Val-hi][Chk]

Hopper answer: [01][00][Dir][00][Chk] ACK or NAK

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

First format message string example for hopper with ser. number (dec 1), to pay out 5 Euro is:

Host sends: [03][05][01][86][01][00][00][F4][01][7B]

Hopper answer: [01][01][03][00][FC] ACK

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

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

[Event Counter]

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

After power down or reset, value of event counter is 0.

[Payout value remaining]

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

[Last Payout: value paid]

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

The value of counter is saved in non-volatile FLASH memory in case if power failure occur during the pay out cycle or after that. It is cleared at the beginning of next pay out command. It shows us how much value has been paid out since last dispense command was launched.

[Last Payout: value unpaid]

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

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

Message format is:

Host sends: [Dir][00][01][85][Chk]

Hopper answer: [01][07][Dir][00][d1][d2-l][d2-h][d3-l][d3-h][d4-l][d4-h][Chk]

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

Host sends: [03][00][01][85][77]

Hopper answer: [01][07][03][00][01][00][00][00][96][00][5E]

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

2.2.5 Command header 132 [hex84], Emergency stop value.

This command immediately halt the payout sequence(*break the motor*) and reports back the value of coins which failed to be paid out. Hopper answer has 100 ms delay for last coin to exit. After Emergency stop value command, the hopper will be disabled.

To perform new payout sequence, hopper must be re-enabled.

Message format is:

Host sends: [Dir][00][01][84][Chk]

Hopper answer: [01][02][Dir][00][d1-l][d1-h][Chk]

Example of message packets for Hopper (address 3) is

Host sends: [03][00][01][84][78]

Hopper answer: [01][02][03][00][7E][04][78]

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

2.2.13 Command header 131 [hex83], Request hopper coin value.

This command return the "name" of specified coin as well as its value.

Coin name is standardized 6 byte ASCII characters.. It is possible to program different coin names that describe the one used in hopper. Unprogrammed coins has code:'-----' .

This code is reserved for unknown(*any type*) of coin.

Message format is:

Host sends: [Dir][01][01][83][01][Chk]

Hopper answer: [01][08][Dir][00][a1][a2][a3][a4][a5][a6][Val-lo]

[Val-hi][Chk]

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

Example of message packets for Hopper(address 3) coin nr. 1(2 Euro) is:

Host sends: [03][01][01][83][77]

Hopper answer: [01][08][03][00][45][55][32][30][30][41][C8][00][BF]

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

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

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

This command show the total number of each type of coins dispensed by hopper.

Each coin counter is non-volatile and has three bytes. LS byte is sent first.

Message format is:

Host sends: [Dir][01][01][82][coin nr.][Chk]

Hopper answer: [01][03][Dir][00][d1][d2][d3][Chk]

Example of message packets for Hopper(address 3) and second coin is:

Host sends: [03][00][01][82][02][54]

Hopper answer: [01][03][03][00][54][00][00][A5]

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

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

2.2.15 Command header 210 [hexD2], Modify sorter paths

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

Host sends: [Dir][02][00][D1][Coin pos.][Sort.Path][Chk]

Hopper answer: [01][00][Dir][00][Chk] ACK or NAK

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

Host sends: [03][02][00][D2][01][02][25]

Hopper answer: [01][00][03][00][FC]

If sorter is not present or path is bigger than the number of paths allowed by the Sorter (2 paths, or 3 paths, or 5 paths), answer will be **NAK**.

Sorter path is initially set to 1 and will return to that value after reset or power down.

2.3 MDCES command headers

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

2.3.1 Command header 253 [hexFD], Address poll

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

Message format is:

Host sends: [00][00][01][FD][Chk] Broadcast message

Hopper answer: **Dly=4x[Address] -> [Address]**

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

Host sends: [00][00][01][FD][02]

Hopper answer: **Dly=12 ms -> [03]** Address is 3

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

Host sends: [00][00][01][FD][02]

Hopper answer: **Dly=1 s -> [FA]** Address is 250

2.3.2 Command header 252 [hexFC], Address clash

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

Message format is:

Host sends: [Dir][00][01][FC][Chk]

Hopper answer: **Random Dly -> [Address]**

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

Host sends: [03][00][01][FC][00]

Hopper answer: **Random Dly -> [03]** Address is 3

3.0 Setting the Hopper Address via PCB DIP-sw

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

Sw 3	Sw 2	Sw 1	Address
Off	Off	Off	3
Off	Off	On	4
Off	On	Off	5
Off	On	On	6
On	Off	Off	7
On	Off	On	8
On	On	Off	9
On	On	On	10

Table 5 Address selection for hoppers AH03-CD.

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

¹For details see ccTalk44-2.pdf, Address poll

²252 bytes of data, source address, header and checksum (total of 255 bytes)

³See Error handling

⁴Refer to Appendix 3.1 of protocol document cctalk43-3.pdf

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**,

D I C H I A R A

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

Modello	Configurazione	N° di Serie e/o matricola
HOPPERCD EVOLUTION	<input type="checkbox"/> CCTALK Mini <input type="checkbox"/> CCTALK Midi <input type="checkbox"/> CCTALK Maxi <input type="checkbox"/> CCTALK Lateral	-----

Essendo realizzato conformemente al modello campione testato con esito positivo ai fini EMC e LVD (rapporto 5500-HEVO.doc del 26 luglio 2010), dalla **STP S.r.l.**, con sede legale in via **Cervese, 373, 47521 Cesena (FC)**, Italia e sede operativa in via **San Donnino, 4, 40127 Bologna (BO)**, Italia, risulta essere conforme a quanto previsto dalle seguenti direttive comunitarie:

- a) le norme armonizzate (per i punti applicabili):
- | | |
|-----------------------------------|-----------------------------------|
| - CEI EN 55014-1 (CEI 110-1); | - CEI EN 61000-3-3 (CEI 110-28); |
| - CEI EN 55014-2 (CEI 210-47); | - CEI EN 61000-4-2 (CEI 210-34); |
| - CEI EN 55022 (CEI 110-5); | - CEI EN 61000-4-3 (CEI 210-39); |
| - CEI EN 55024 (CEI 210-49); | - CEI EN 61000-4-4 (CEI 210-35); |
| - CEI EN 60065 (CEI 92-1); | - CEI EN 61000-4-5 (CEI 110-30); |
| - CEI EN 60335-1 (CEI 61-150); | - CEI EN 61000-4-11 (CEI 110-29); |
| - CEI EN 60335-2-82 (CEI 61-226); | - CEI EN 61000-6-1 (CEI 210-64); |
| - CEI EN 60950-1 (CEI 74-2); | - CEI EN 62233 (CEI 61-251). |
| - CEI EN 61000-3-2 (CEI 110-31); | |
- 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, ___/___/_____

Fabrizio Alberici

Il Presidente

Alberici S.P.A.

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

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

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

E-mail: info@alberici.net – Url: <http://www.alberici.net>



NOTA: La Alberici S.p.A. si riserva il diritto di apportare modifiche alle specifiche tecniche dell'apparecchiatura descritta in qualunque momento e senza preavviso, nell'ambito del perseguimento del miglioramento continuo del proprio prodotto.



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
ITALY

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

<http://www.alberici.net>

info@alberici.net