HopperOne S11 ccTalk

Standard / Reverse

Operator's Manual

A.u.S. Spielgeräte GmbH
Scheydgasse 48, AT 1210 Wien

Open times: Mo. - Fr. 9-18
Tel. +43 1 271 66 00 66 - Fax. +43 1 271 66 00 75
E-Mail verkauf@aus.at - Web www.aus.at

1.General description

Congratulations for having purchased of our **HopperOne S11!** This HopperOne S11 has been designed and realized in the **Alberici's research laboratories** and fulfilss all the requirements of the coin-op market. This belt drive single denomination dispensing device makes use of the most modern electronic and mechanical technologies. It is secure, enduring, reliable.

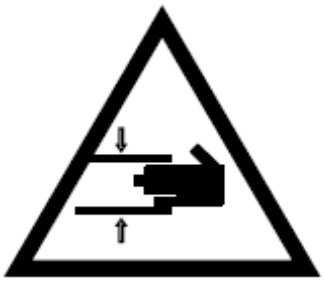
It is available either with traditional **electrode sensors (P Model)** or with **optics level sensors (O Model)**; the latter type allows to eliminate all malfunctionings due to ESD accumulated by the coins.

1.1 Sphere of use

The technology implemented makes the **HopperOne S11** able to manage different operations, i.e. to count up and control the type of coins paid out, and to stop automatically when empty. To this purpose it makes use of a significant quantity of control routines for the management of the internal and external events.
It integrates easily into **Gaming and Slot machines, Money Changers, Kiosks and Vending Machines**.
These features make it easily compatible with all the cards normally available on the market.

1.2 Safety

The HopperOne S11 can 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.



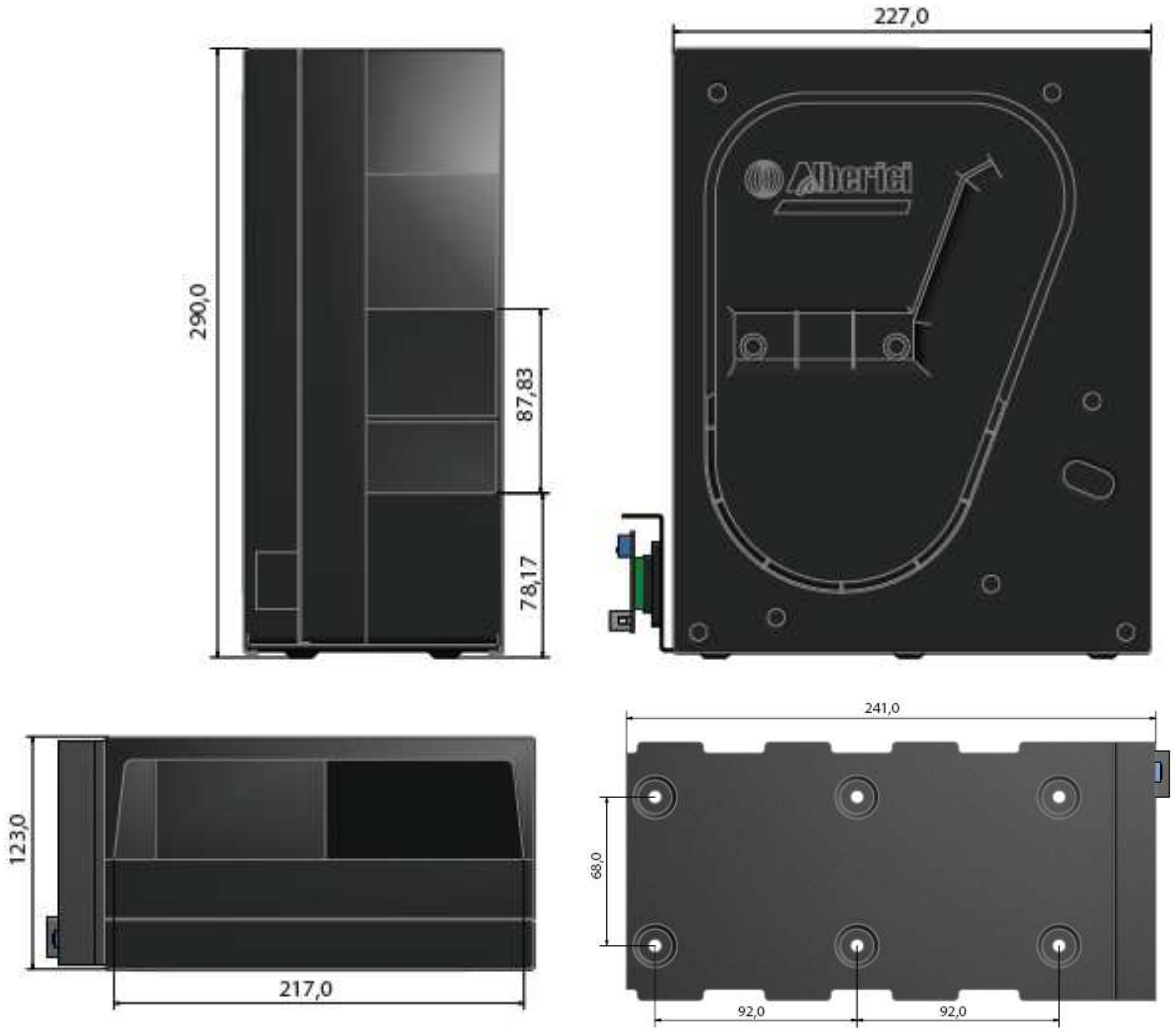
ATTENTION:
HARM DANGER!

MECHANICAL PARTS IN MOTION

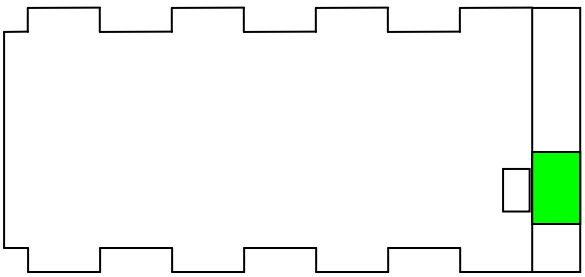
2. Mechanical description

The HopperOne S11 cctalk is available in 2 different versions, according to the respective positions of the electrical connector and the coins outlet. When they are located at opposite sides, the version is named "STANDARD"; when they are located at the same side, the version is named "REVERSE".
The standard features of the HopperOne S11 make it interchangeable with similar devices already existing in the market. It can handle any coins whose diameter ranges between 16 mm and 32 mm (choose the most convenient belt for your purpose: 16-24mm, or else 22-32mm). Coin thickness can range between 2,0 mm and 3,4 mm.

2.1 Overall dimensions

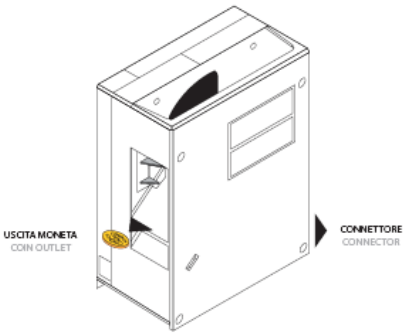


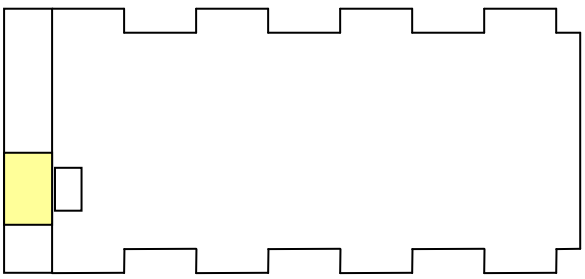
2.1 Position of the Chinch connector



Reverse version

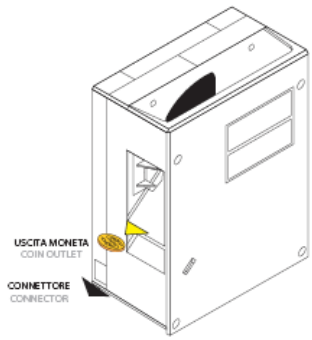
(connector on the same side of the output of the coins)





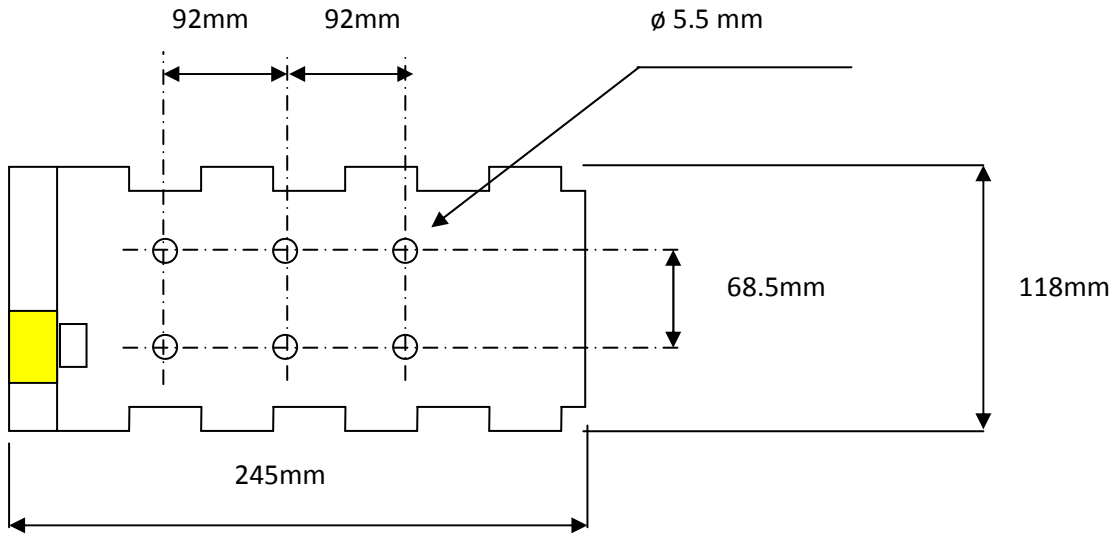
Standard version

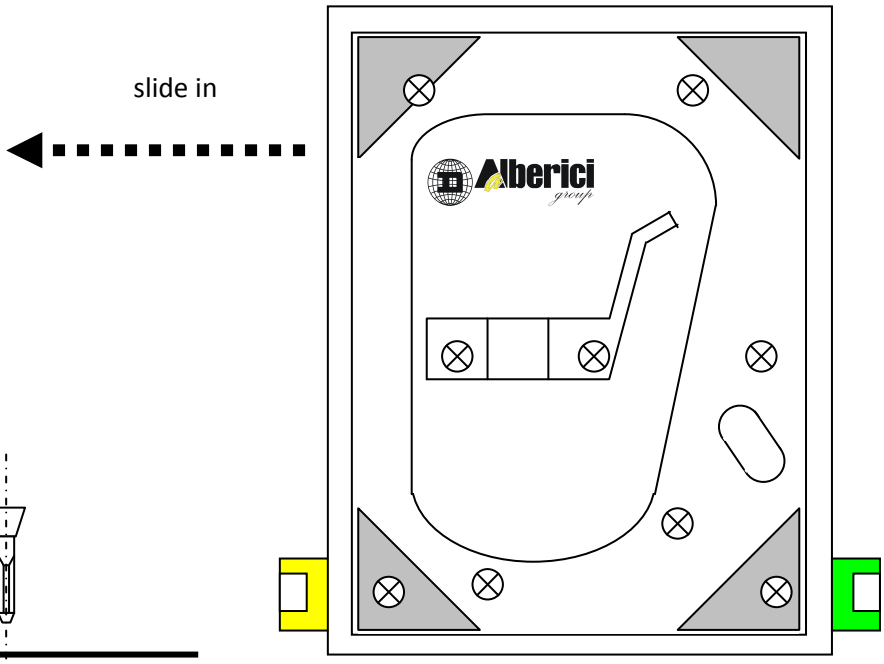
(connector on the opposite side of the output of the coins)



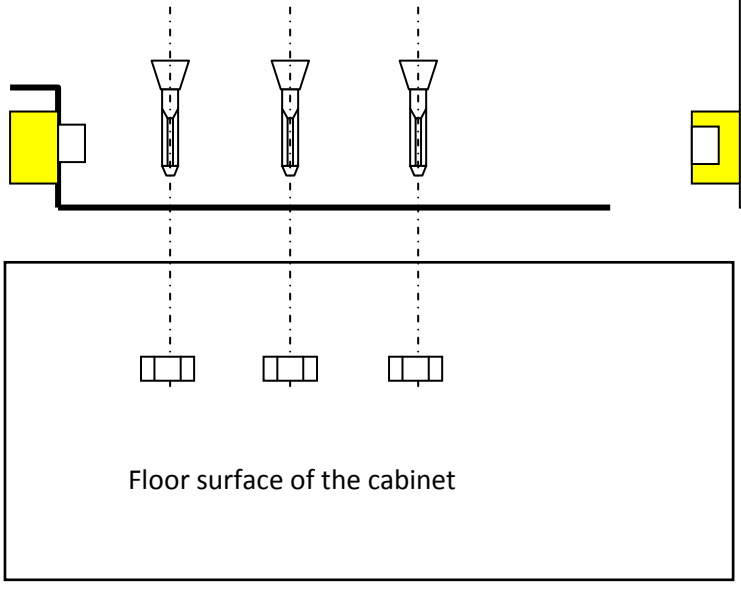
2.3 Installation

. *fasten the slide support,*
. *slide the hopper in*
. *for electrical connections , please see chapter 3*





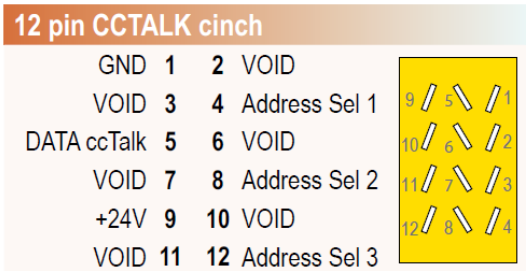
Please use conical head bolts

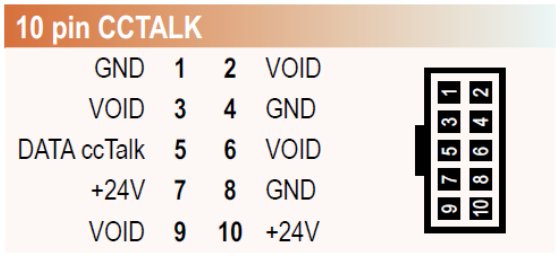


3. Electrical description

All the signals handled by the hopper are **in negative logic**: each signal is considered active when it is LOW (GND).

3.1 Connector pin-out





The connector can be located on the opposite side (**Standard version)** of the coins outlet, or on the same side **(Reverse version).**

⚠ *PLEASE NOTE: the pc board of the model with plate level sensors is different from the pc board of the model with optic level sensors.*

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

3.2 Power Supply

This equipment must be supplied with direct voltage.
The **+24Vdc** to **pin9** (**= pin 7** on the 10-pin connector) are used for both the operation of the logic card and of the motor.
Max. Payout speed is 240 pcs. output per minute.

Current draw:

		Stand-by	No load	Normal operation	Stuck (*)
Board	+12Vdc	40mA/0.48W	40mA/0.48W	80mA/0.96W	80mA/0.96W
Motor	+24Vdc	0mA/0m W	70mA/1.4W	1.2 A/28.8W	1.5mA*/30W
Total		0.48 W	110mA/1.88W	29.76W	30.96W

*The motor overload current draw shall always be limited by the electronic circuit. The 1 A draw, corresponding to hold-up of the motor, will therefore be reached only for a few msec.

Technical data:

Operating Voltage	24 Vdc	**Max current draw**	1 A
Protocol	ccTalk	**Stand-by current draw**	40 mA
Speed	220 coins/minute	**Operating current draw**	400 mA
Capacity	1200 coins(Φ24mm)	**Operating temperature**	0 °C - 50 °C
Coin diameter	16-32 mm	**Operating Humidity**	20% - 75% non condens.
Coin thickness	2-3,4 mm	**Weight**	2,2 Kgs

ccTalk communication protocol

cctalk® communication protocol is the Money Controls[1] 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.
The communication protocol of the Alberici ccTalk HopperCD is implemented according to generic specification 4.2
Serial communication was derivated from RS232 standard. Low data rate NRZ (***N**on **R**eturn to **Z**ero*) asyncronous communication: Baud rate 9600, 1 start bit, 8 data bits, no parity, 1 stop bit. RS232 handshaking signals (*RTS, CTS, DTR, DCD, DSR*) are not supported. Message integrity is controlled by means of checksum calculation.

1 Communication specifications

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

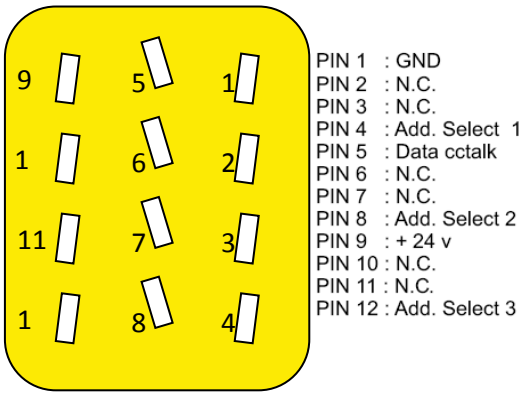
To reduce the costs of connections the “Level shifted “ version of RS232 is used. The idle state on serial connector is 5V, and active state is 0V.

Mark state (*idle*) +5V nominal from 3.5V to 5V Space state (active) 0V nominal from 0.0V to 1.0V

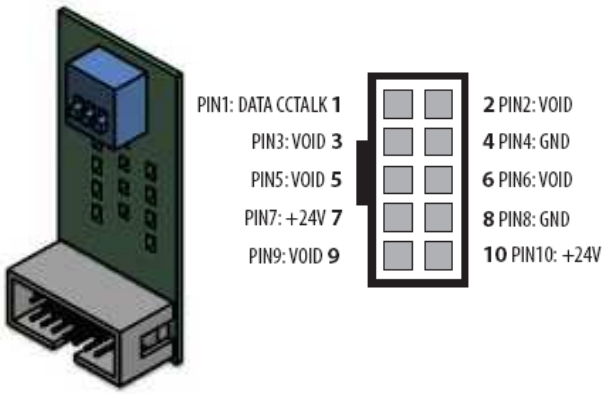
Data I/O line is “open collector” type, so it is possible to use device in systems with different voltage (*12V pull up in older devices*).

1.3 Connection

The connection of HopperOne S11 to the machine circuit is made through the Chinch connector. It can be made simpler by means of the optional 10-pin connector, handling both the power supply and the communication data.



PIN 1 : GND
PIN 2 : N.C.
PIN 3 : N.C.
PIN 4 : Add. Select 1
PIN 5 : Data cctalk
PIN 6 : N.C.
PIN 7 : N.C.
PIN 8 : Add. Select 2
PIN 9 : + 24 v
PIN 10 : N.C.
PIN 11 : N.C.
PIN 12 : Add. Select 3



[1] Formally Coin Controls

1.4 Message structure

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

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

There is an exception of message structure when device answer to instruction Address poll and Address clash[2]. The answer consists of only one byte representing address delayed for time proportional to address value. For CRC checksum case format is:

[Destination address]
[Nr. of data bytes] [
CRC 16 LSB] [Header
] [Data 1] ... [Data n] [
CRC 16 MSB]

1.4.1 Address

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

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

Table 1 Standard address for different types of devices

Address for Alberici HopperOne S11 is factory-set at value 3, but the user can change the default address using MDCES instructions Address change or Address random or setting Hopper external dip-switch.

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

1.4.2 Number of data byte

Number of data byte in each transfer could be from 0 to 252. Value 0 means that there are no data bytes in the message, and total 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[3].

1.4.3 Command headers (*Instructions*)

Total amount of possible cctalk command header is 255 with possibility to add sub-headers using headers 100, 101, 102 and 103. **Header 0** stands for **ACK** (***ack****nowledge*) replay of device to host. **Header 5** stands for **NAK** (***N****o* ***ack****nowledge*) replay of device to host. **Header 6** is **BUSY** replay of device to host. In all three cases no data bytes are transferred. Use of ACK and NAK headers 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

Alberici Hopper use 24[4] instructions-headers.
Details are explained in chapter 2.

1.4.4 Data

There is no restrictions data formats use. Data could be BCD (***B****inary* ***C****oded* ***D****ecimal*)numbers, Hex numbers or ASCII strings. Interpretation as well as format is specific to each header use, and will be explained in separate chapter.

1.4.5 Checksum

Message integrity during transfer is checked by use of simple zero checksum calculation. Simple checksum is made by 8 bit addition (modulus 256) of all the bytes in the message. If message is received and the addition of all bytes are non-zero then an error has occurred[5]. For noisy environment or higher security application it is possible to use more complex, 16 bit CRC CCITT checksum based on a polynomial of: $\mathbf{x^{16} + x^{12} + x^{5} + 1}$ and initial value of CRC register **0x0000**.

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

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

1.5 Timing specification

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

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 msec. 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. HopperOne S11 Command header set

Table 2 shows the Command header set that host can use in communication with Hopper (*)

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

(*) As option, fw version v. A1.02 can be requested.
This one contains Header 25 (Hex19), that commands complete hopper emptying.

Command headers are divided in to 3 different groups:

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

2.1 Common command headers

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

2.1.1 Command header 254 [hexFE], Simple poll

The fastest way for host to detect all attached devices in cctalk network. Addressed device -Hopper answer with ACK (*Acknowledge*). If within predicted amount of time Hopper does not answer, probably is not connected, powered or simple not working properly. Message format is:

Host sends: **[Dir] [00] [01] [FE] [Chk]** Hopper answer: **[01] [00] [Dir] [00] [Chk]** Hopper default address is 3, example of message packet is:

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

2.1.2 Command header 246 [hexF6], Request manufacturer ID

Hopper answer with ASCII string representing manufacturer name.
Message format is:

Host sends: **[Dir] [00] [01] [F6] [Chk]** Hopper answer: **[01] [Nr.b] [Dir] [00] [a1] [a2] . . . . [an] [Chk]**
Nr. b is number of data bytes-characters sent by Hopper, and a1 to an are ASCII characters.
For **Alberici group** Hopper, example of message packet is:

Host sends: **[03] [00] [01] [F6] [06]** Hopper answer **[01] [0E] [03] [00] [41] [6C] [62] [65] [72] [69] [63] [69] [20] [67] [72] [6F] [75] [70] [86]**

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

Answer to command header is standardized name for Hopper. It answer with ASCII string of characters representing standardized name for that type of device **Payout**. Message format is:

Host sends: **[Dir] [00] [01] [F5] [Chk]** Hopper answer: **[01] [06] [Dir] [00] [50][61][79][6F][75][74][Chk]**
Number of data byte is always 6, hex [06]. Example of message packets for coin selector (*address 3*) is:

Host sends: **[03] [00] [01] [F5] [07]** Hopper answer: **[01] [06] [03] [00]] [50][61][79][6F][75][74] [74]**

2.1.4 Command header 244 [hexF4], Request product code

Hopper answer with ASCII string of character, representing its factory type. For Alberici Hopper it's **HopperTwo ccTalk**. Message format is:

Host sends: **[Dir] [00] [01] [F4] [Chk]** Hopper answer: **[01] [10] [Dir] [00] [a1][a2] . . . [an] [Chk]** Number of data bytes sent by Hopper is 16, hex [10]. Example of message packets for Hopper (*address 3*) is :

Host sends: **[03] [00] [01] [F4] [08]**
Hopper answer: **[01][10][03][00][48][6F][70][65][72][54][77][6F][20][63][63][54][61][6C] [6B][D2]**

2.1.5 Command header 242 [hexF2], Request serial number

Hopper answer with three byte serial number.
Message format is:

Host sends: **[Dir] [00] [01] [F2] [Chk]** Hopper answer: **[01] [03] [Dir] [00] [Serial 1 -LSB] [Serial 2] [Serial 3 -MSB] [Chk]** Serial 1 – first data byte sent is LSB of serial number. Example of message packets for Hopper (*address 3*) and serial number **1-2-34567,** hex [BC][61][4E] is:

Host sends: **[02] [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 its own system of labeling. Example of message packets for Hopper (*address 3*) is:

Host sends: **[03] [00] [01] [F1] [0B]** Hopper answer: **[01] [04]**
[03] [00] [31] [2E] [32] [31] [36] Hopper answer is '**1.21'**.

2.1.7 Command header 192 [hexC0], Request build code

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

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

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

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

[6] Usually label printed on electronic circuit board

2.1.8 Command header 169 [hexA9], Request address mode

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

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

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

Host sends: **[03] [00] [01] [A9] [53]** Hopper answer: [01] [01] [03] [00] [B7] [44] Hopper answer with data [B7]. It means that address may be changed with serial command (non volatile).If answer is [B3], mean that address is selected via interface connector.

2.1.9 Command header 4 [hex04], Request comms revision

.

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

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

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

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

2.1.10 Command header 1 [hex01], Reset device

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

[7] Details of description see in public document cctalk42-2.pdf

[8] Address change, Address random

2.2 Hopper command headers

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

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

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

Host sends:	**[Dir] [04] [01] [DB] [PIN1-LSB][PIN2][PIN3][PIN4-MSB] [Chk]**
Hopper answers:	**[01] [00] [03] [00] [FC]** ACK if PIN is correct
Hopper answers:	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 msec. 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

[9] See next chapter

BIT4 -Low level sensor support 0 –
Features not supported or fitted 1 -
Features supported and fitted

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

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

Message format is:

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

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

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

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

Hopper answer: **[01] [01] [03] [00] [31] [CA]**

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

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

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

Host sends: **[Dir] [00] [01] [D8] [Chk]**

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

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

Host sends: **[03] [00] [01] [D8] [24]**

Hopper answer: **[01] [05] [03] [00] [00][00][00][00][00] [F7]**

2.2.4 Command header 172 [hexAC], Emergency stop.

This command immediately halt the payout sequence and reports back the number of coin 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 dispense coin from the hopper. Maximum number of coin hopper can dispense with a single command is 255. After 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][Ch k]** Hopper

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

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

Host sends: **[03] [04] [01] [A7] [12] [34] [56] [64][Chk]** Hopper

answer: **[01] [00] [03] [05] [F7] NAK**

Command try to pay 100 coins (64H) but serial number sent to hopper isn't correct.
Second command format is

Host sends: **[Dir][0A][01] [A7] [00] [00] [00] [00] [00] [00] [00] [00] [00] [N°Coin][Chk]** Hopper

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

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

Host sends: **][09][01] [A7] [00] [00] [00] [00] [00] [00] [00] [00] [01][4B]**
Hopper answers: **[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:
1. Event Counter that show the number of good dispense events since last reset.
2. Payout coins remaining that show how many coins are still to pay.
3. Last Payout: coins paid, that show how many coins paid out since last dispense command (increments with each coin dispensed)
4. 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 error. Bit meaning is shown here :

BIT0 – Absolute maximum current exceeded BIT1 – Payout timeout occurred BIT2 – Motor reverse during last payout to clear a jam BIT3 – Opto fraud attempt, path blocked during idle BIT4 – Opto fraud attempt, short circuit during idle BIT5 – Opto blocked permanently during payout BIT6 – Power up detected BIT7 – Payout disabled

Command format is

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

Hopper answer: **[01] [00] [Dir] [00] [d1] [d2] [Chk]**

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

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

Hopper answer: **[01] [02] [03] [00] [C0] [00] [3A]**

The data byte Hex[60] mean that Opto sensors 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 answers: **[01] [00] [00] [00] [FD]** ACK *Address is changed*

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

A.u.S. Spielgeräte GmbH
Scheydgasse 48, AT 1210 Wien

Open times: Mo. - Fr. 9-18
Tel. +43 1 271 66 00 66 - Fax. +43 1 271 66 00 75
E-Mail verkauf@aus.at - Web www.aus.at