

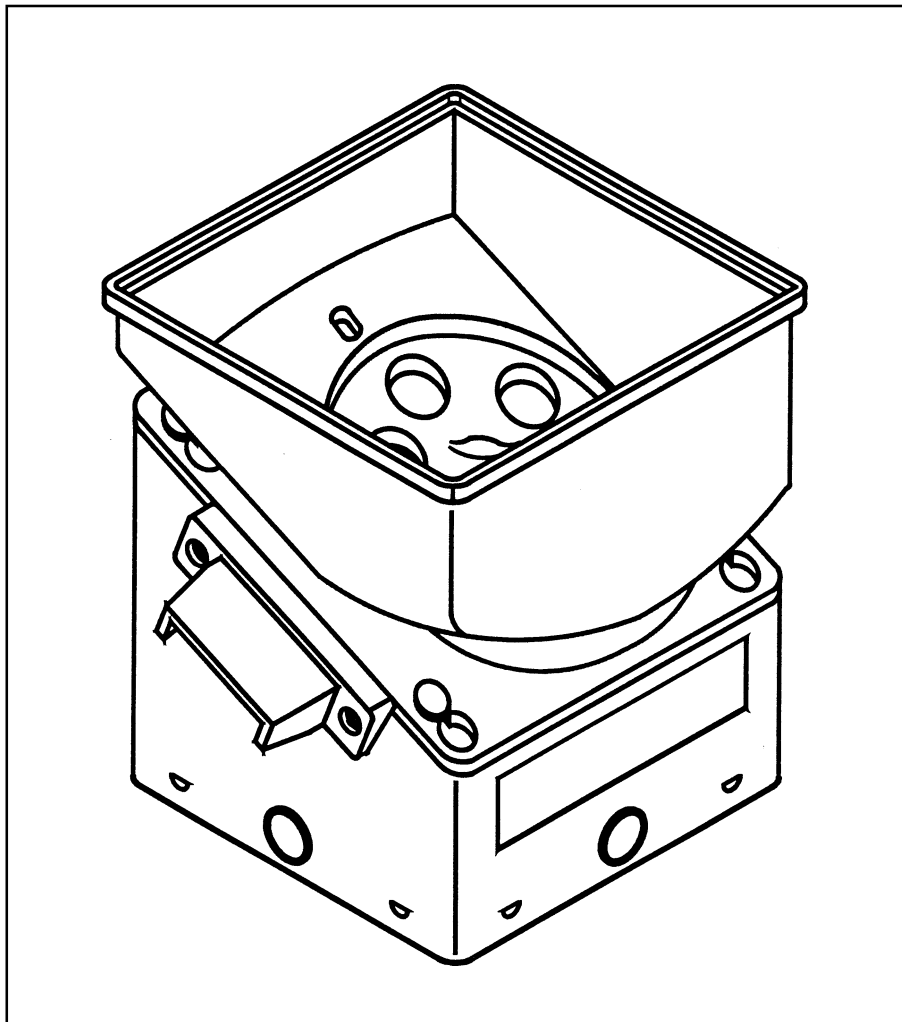
# CUBE HOPPER<sup>®</sup> MKII

## Product Manual

**Model 10-1900-XX  
(model with cctalk interface)**

Version 0.6 / April 2004

A new design to solve most of your payout problems!



Serving the coin-machine industries since 1955

## **Attention!**

Always turn power off before removing or installing the hopper in order to prevent any damage due to surge currents!

## Revision History

Revision	Date	Comment	By
0.4	20 oct 2003	<ul style="list-style-type: none"> <li>- Document is adapted to Cube Hopper Firmware V1.9</li> <li>Versions 1.4 - 1.8 are versions used for test and evaluation of the cctalk implementation on the cube hopper.</li> <li>- New Appendix added: CCTalk Cube Hopper Test Program.</li> <li>- Maximum hopper voltage set from 32Vdc to 26Vdc.</li> </ul>	R.T.
0.5	7 jan 2004	- Minor text changes.	R.T.
0.6	12 april 2004	- New PC Test program V1.10	R.T.

## Contents

Tables.....	5
Figures.....	5
1. Product description.....	6
1.1 Features of the cctalk Cube Hopper .....	6
Serial Communication: More control over the devices .....	8
Serial Communication: Fraud elimination? .....	8
Serial Communication: Hopper responsibilities.....	8
Serial Communication: Bookkeeping.....	8
2. CCTalk implementation on Suzo Cube Hopper .....	9
2.1 Serial Connector Pinout .....	9
2.2 Interface circuit .....	10
2.3 Device Address.....	10
3. Commands.....	11
3.1 Command / Response frame format.....	12
3.2 Hopper Setup and Initialisation commands.....	12
Simple Poll (header 254) .....	12
Address Poll (header 253) .....	12
Request comms revision (header 004) .....	13
Request comms status variables (header 002) .....	13
Clear comms variables (header 003) .....	13
Reset device (header 001) .....	13
Request Manufacturer id (header 246) .....	13
Request Equipment Category id (header 245) .....	13
Request Product Code (header 244) .....	14
Request Serial Nr (header 242) .....	14
Request Software Revision (header 241).....	14
Request data storage availability (header 216) .....	14
Request buid code (header 192).....	15
Request address mode (header 169).....	15
Enter PIN number (header 218).....	15
Enter New PIN number (header 219).....	15
Modify variable settings (header 165) .....	16
Request variable settings (header 247).....	16
3.3 Hopper Dispense coins procedure.....	17
Test Hopper (header 163) .....	17
Request hopper status (header 166).....	18
Enable Hopper (header 164) .....	19
Pump RNG (header 161).....	19
Request Cipher Key (header 160) .....	19
Dispense Hopper coins (header 167).....	20
Request Hopper Status (header 166) .....	21
Request Payout high/low status (header 217) .....	21
Request Hopper dispense count (header 168).....	22
Read Data Block (header 215) .....	22
Write Data Block (header 214).....	22
Summary dispense coins example: .....	23
Host program example (only used for protocol explanation) in pseudo code.....	24
3.4 Coin Jams during payout.....	26
3.5 Power failures .....	26
Appendix 1. PC Interface Circuit.....	28
Appendix 2: Mechanical dimensions .....	29
Appendix 3. Maintenance .....	30
Removing the Motor and Gearbox .....	30
Removing and re-installing the hopper.....	30
Cleaning and Materials.....	30
Disassembly .....	31
How to remove the disc.....	31

---

How to remove the coin insert plate.....	31
How to re-configure the Cube Hopper.....	31
Appendix 4: TroubleShooting .....	32
Coins fail to unjam .....	32
Motor fails to run.....	32
Over payout of coins.....	32
Under payout of coins.....	32
CCTalk communication trouble shooting .....	32
Appendix 5: Optional add-ons.....	33
Cup extension (Part No. 10-0200).....	33
Appendix 6: Payout discs and inserts .....	34
Appendix 7: Reference chart .....	35
THE "EURO-HOPPERS" .....	36
Appendix 8: Exploded view.....	37
Appendix 9: CCTalk Cube Hopper Test Program.....	38
Installation .....	38
Example1: Payout 3 coins.....	39
Step 1: Transmit Hopper Test command.....	39
Step 2: Enable Hopper .....	40
Step 3: Get Serial Number from the Hopper.....	41
Step 4: Transmit Dispense command .....	42
Example2: Full Hopper Test .....	44
Section Hopper Data .....	45
NV Data section .....	46
Device Data section .....	46
Network Traffic section .....	46
Acknowledgement .....	46

## Tables

Table 1: Connector pinout .....	9
Table 2: Address select pins.....	10
Table 3: Hopper commands .....	11
Table 4: Hopper Product Codes.....	14
Table 5: Address mode .....	15
Table 6: Hopper Variable Settings .....	16
Table 7: Hopper Status Register 1.....	17
Table 8: Hopper Status Register 2.....	17
Table 9: LevelStatus bit definition .....	21
Table 10: Electrical specifications.....	27
Table 11: Timing specifications.....	27
Table 12: EEPROM Memory Description .....	27
Table 13: Payout discs .....	34
Table 14: Coin insert plates.....	34

## Figures

Figure 1: Traditional machine wiring .....	7
Figure 2: Network wired machine .....	7

## 1. Product description

The CUBE HOPPER is a single coin payout system made universal by using the relevant payout disc and coin insert plate (See Appendix 6: Payout discs and inserts).

It can be used with all round coins with a diameter between 18.00 and 31.00mm and a thickness ranging from 1.50 to 3.20mm.

The hopper is square and easy to mount by means of a standard mounting-bracket (included).

By using a unique “sun and planet” gearbox system this hopper offers the possibility to contain and count up to 1,000 coins of € 0.50 (if fitted with two cup-extensions).

The hopper is protected by an auto-reverse anti-jam system and has an indirect optical readout.

The Cube Hopper Mk2 from Suzo International is available with a cctalk interface.

The purpose of this document is not to describe the cctalk protocol, only those aspects concerning the operation of the cube hopper are mentioned.

Refer to the following documents from MoneyControls for a complete cctalk specification:

- cctalk Serial Communication Protocol, Generic Specification, Issue 4.2

### 1.1 Features of the cctalk Cube Hopper

- wide power supply voltage range: hopper runs on any voltage between 12 – 24 Volt.
- excellent anti-jam performance.
- full bridge MosFet motor drive (no mechanical motor switch).
- Pulse Width Modulated (20kHz) motor control giving:
  - constant pay out speed control (at 24VDC)
  - low motor start-up current, prevents systems power-dips.
- secured payout commands.
- continuous opto-sensor check.
- anti-jam operation prevents hopper blocking.
- low level sense plates are standard on the cube hopper.
- high level sense pin optional.
- software is watch-dog protected.

## 1.2 Serial Communication: Why and how to use it

This section describes some differences between a conventional build machine and a machine using a network like cctalk.

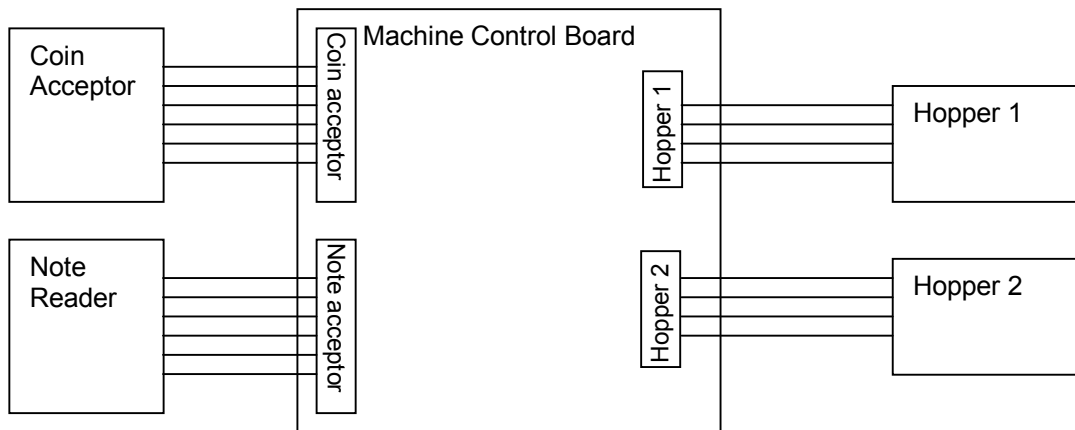
### Serial Communication: One (cheap) network cable does it all

Traditional build machines have a central control board that controls all attached devices.

Each hopper is connected to the board with it's own 4 wire cable and connectors.

Money acceptors are connected to the board using flat cables of 10 wires or more with their own connectors.

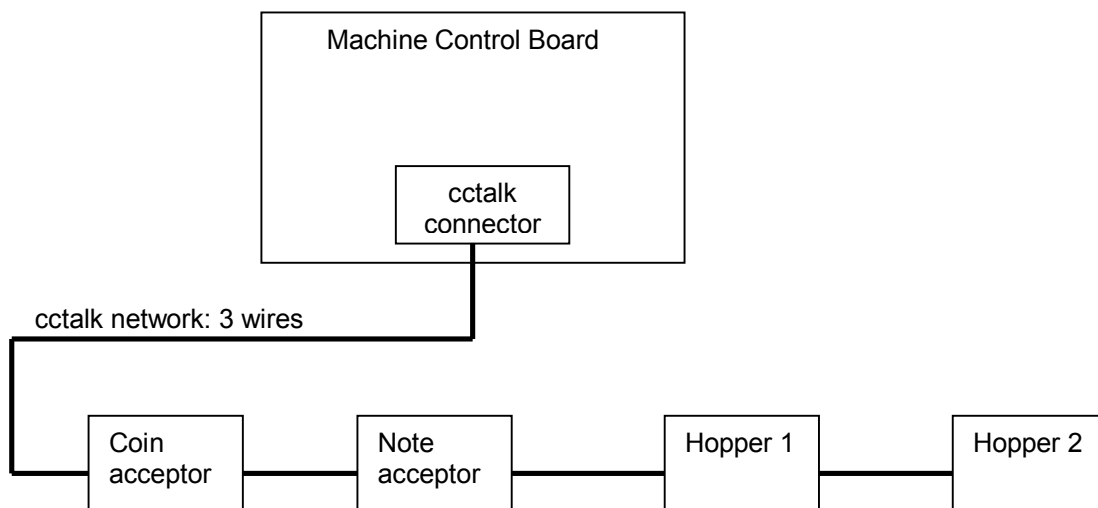
See Figure 1: Traditional machine wiring



**Figure 1: Traditional machine wiring**

From fig. 1 can be seen that you need often 3 different cables (1 coin acceptor, 1 note acceptor and 2 hoppers) in order to build a system. This is a quite expensive system.

Figure 2 shows how a system looks if a cctalk network is used.



**Figure 2: Network wired machine**

As can be seen from figure 2, a networked solution is much simpler, because only 1 long cable consisting of 3 wires (power, data, ground) connect all devices to the control board. Each device would typically use the same connector.

#### **Serial Communication: More control over the devices**

The machine controls the devices by sending data over the network, just like the famous RS232 communication network. The data consists of bytes that are grouped together to form cctalk messages. Each device has its own set of command messages and has its own unique cctalk address. If the machine sends a message to a hopper, it will put the hopper's address into the message and send it over the network to the hopper.

The cctalk protocol defines all messages that are used to communicate with all devices.

There are messages to start the hopper, stop the hopper, return coin level, etc.

So much more control over each device is achieved by using device control messages.

#### **Serial Communication: Fraud elimination?**

The traditional way of hopper fraud would be to drill a hole into the machine cabinet and put 24V on the proper wire of the hopper cable.

With a network, each device is controlled using serial data communication. Now you need a PC and knowledge about the cctalk command messages to start a hopper.

In order to make hopper fraud very difficult, some hopper manufacturers like MoneyControls use sophisticated encrypted pin codes in their commands in order to start a hopper. Others like Azkoyen use simple codes in their commands to start a hopper.

#### **Serial Communication: Hopper responsibilities**

Each cctalk hopper is equipped with a microcontroller implementing the cctalk protocol.

You have to define exactly what the hopper should do when it receives a command.

If the hopper receives a payout command, the hopper should start a payout only if the pincode is correct. When it has paid the requested amount of coins it should stop automatically. It is the responsibility of the hopper to stop when the requested amount of coins is paid.

#### **Serial Communication: Bookkeeping**

Some hoppers also count the number of coins paid and the number of coins unpaid and even have short term and long term running counters. However is it practical to store all these numbers into the hopper? The machine itself also has its own copy of all running counters. How do you maintain hopper data and machine data consistent? What happens with the data during a system reset or a power down event? MoneyControls retain their counter data if a Reset command is received whereas Azkoyen clears all counters. What about a power down cycle? MoneyControls clears the coins unpaid counter if the hopper was not running during a payout, whereas Azkoyen does not. MoneyControls retain the short term and long term counter data, whereas Azkoyen only retains the last status values. Suzo hopper V1.9 does support the bookkeeping of MoneyControls. But in general only the number of coins paid is the one that is most practical. If the hopper is commanded to pay 10 coins and a power down event occurs, then after the power has returned, the host machine should check how many coins the hopper has paid just before the power failed. The machine should check this number with its own records and take appropriate action.

For hopper maintenance, the total number of coins dispensed during its life would also be a useful number.

## 2. CCTalk implementation on Suzo Cube Hopper

The protocol conforms to cctalk b96.p0.v12.a5.d0.c8.m0.x8.i1.r4 and b96.p0.v24.a5.d0.c8.m0.x8.i1.r4.

9600 baud  
 open-collector  
 +12V .. +24V nominal supply  
 +5V data pull-up  
 supply sink  
 connector type 8  
 slave device  
 8-bit addition checksum  
 no encryption  
 cctalk minor release 1  
 cctalk major release 4

### 2.1 Serial Connector Pinout

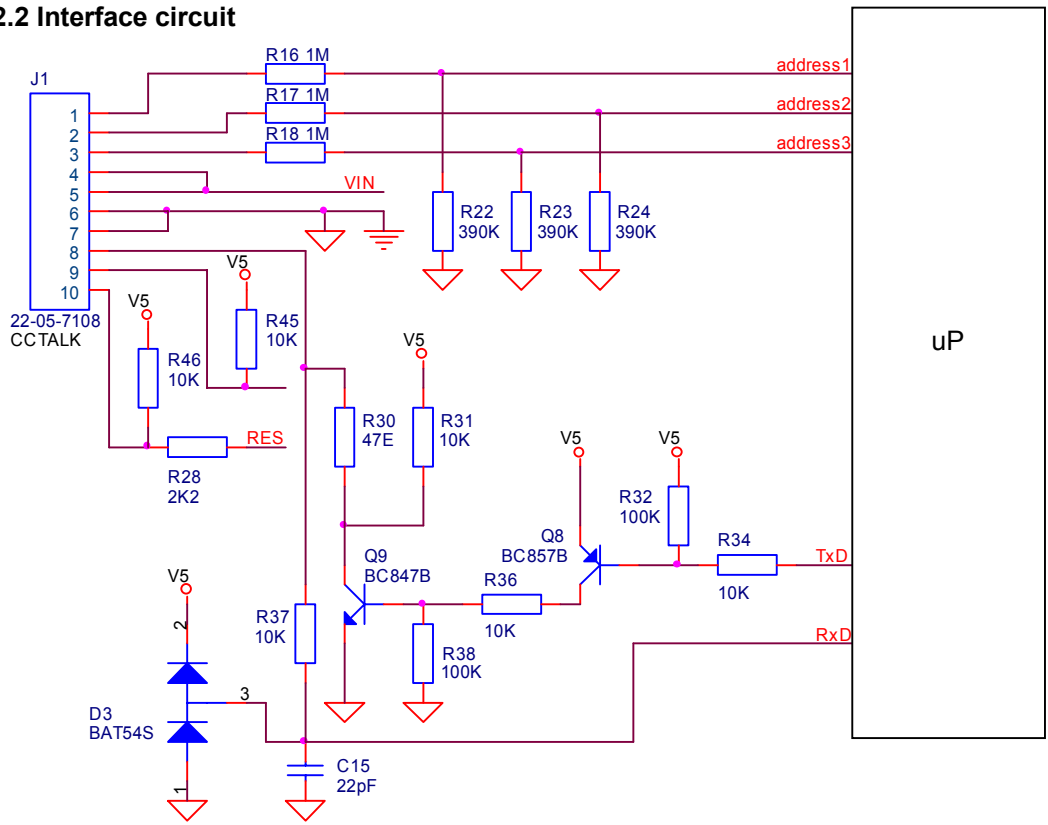
Connector type: Molex 22-05-7106 header with friction lock.  
 The following socket connector can be used from Molex: 22-01-1103.

Pin	Function
1	Address select 3 - MSB
2	Address select 2
3	Address select 1 – LSB
4	+Vs
5	+Vs
6	0V
7	0V
8	/DATA (cctalk)
9	Hopper High Level Sense
10	/RESET

**Table 1: Connector pinout**

Pins 4, 5 are linked together and pins 6,7 are linked together, and can be used to daisy-chain the power wires from hopper to hopper.

### 2.2 Interface circuit



### 2.3 Device Address

All Cube Hoppers leave the factory with **address 3**. Other addresses can be set by wiring the address select lines. See Table 2.

The address can be changed with serial commands. Unless you have an application requiring more than one cube hopper on the serial bus, it is strongly recommended you leave the address alone. The default addresses for coin acceptors and bill validators have been made different and will not clash with the cube hopper. When the hopper is powered up or reset, the hopper will always revert back to its physical address. Other addresses can be set by commands.

For applications requiring more than one hopper on the serial bus, one or more of the address select lines may be connected to +Vs. A total of 8 unique bus addresses may be generated in this way, in the range 3 to 10 inclusive.

X = Connect to +Vs (Pin 4, 5)			Serial Address
Address select 3	Address select 2	Address select 1	
			3
		X	4
	X		5
	X	X	6
X			7
X		X	8
X	X		9
X	X	X	10

**Table 2: Address select pins**

### 3. Commands

The following cctalk commands are currently implemented in the cctalk cube hopper.

Header	Data bytes	Response (default)	Description
254	<none>	ACK	Simple poll
253	<none>	{variable delay} [slave address]	Address poll
252	<none>	{variable delay} [slave address]	Address clash
251	[new address]	ACK	Address change
250	<none>	ACK	Address random
247	<none>	[current_limit] [motor stop delay] [payout timeout] [max current measured] [supply voltage] [connector address]	Request variable set
246	<none>	"Suzo Int (NL)"	Request manufactur id
245	<none>	"Payout"	Request equipment category id
244	<none>	"SCH2" "SCH2-NOENCRYPT" 'SCH2-USE_SERNR'	Request product code (Respondse depends on factory settings)
242	<none>	[serial 1- LSB] [serial 2] [serial 3 – MSB]	Request serial number
241	<none>	"Cube V1.3"	Request software version
236	<none>	[opto state] (bit 7 set: path blocked)	Read opto states
219	[PIN1] [PIN2] [PIN3] [PIN4]	ACK	Enter new PIN number
218	[PIN1] [PIN2] [PIN3] [PIN4]	ACK	Enter PIN number
217	<none>	[level status] (bit 0 set: low level)	Request payout level status
216	<none>	[memory type] [read blocks] [read bytes per block] [write blocks] [writes per block]	Request data storage availability
215	[block number]	[data 1] [data 2] ... [data 8]	Read data block
214	[block nr] [data 1] ... [data 8]	ACK	Write data block
192	<none>	"Lev Lo"	Request build code
172	<none>	[payout coins remaining]	Emergency stop
171	<none>	"-----"	Request hopper coin
169	<none>	[address mode]	Request address mode
168	<none>	[nr coins 1 – LSB] [ nr coins 2] [nr coins 3 – MSB]	Request hopper dispense count
167	[sec 1] [sec 2] [sec 3] [sec 4] [sec 5] [sec 6] [sec 7] [sec 8] [N coins]	[event counter]	Dispense hopper coins
166	<none>	[event counter] [payout coins remaining] [last payout: coins paid] [last payout: coins unpaid]	Request hopper status
165	[current limit] [motor stop delay] [payout TO] [max current measured] [supply voltage] [connector address]	ACK	Modify variable set
164	[enable code]	ACK	Enable hopper
163	<none>	[hopper status register1] [hopper status register 2]	Test hopper
161	[rnd 1] [rnd 2] [rnd 3] [rnd 4] [rnd 5] [rnd 6] [rnd 7] [rnd 8]	ACK	Pump RNG
160	<none>	[key 1] [key 2] [key 3] [key 4] [key 5] [key 6] [key 7] [key 8]	Request cipher key
004	<none>	[cctalk level] [major revision] [minor revision]	Request comms revision
003	<none>	ACK	Clear comms status variables
002	<none>	[rx timeouts] [rx bytes ignored] [rx bad checksums]	Request comms status variables
001	<none>	ACK	Reset device

**Table 3: Hopper commands**

A detailed explanation of all commands follows in the next sections.

### 3.1 Command / Response frame format

Dest. Addr	Nr Data Bytes	Source Addr	Header	Data 1	...	Data N	Checksum
------------	---------------	-------------	--------	--------	-----	--------	----------

The destination address (host address) is usually 1.  
 The source address (hopper address) starts from 3.

Most responses from the slave have a ACK Header byte followed by zero or more data response bytes. If a command cannot be executed, a NAK Header (Hex 5) will be returned. For example the hopper dispense command which may return a NAK (Hex 5) if the dispense procedure could not be done (for example hopper disabled, busy, pin code not correct, opto-error, etc).

The ACK byte in the responses has value 0.  
 The Checksum is calculated such that the 8-bit addition (modulus 256) of all bytes in the message from the start to the checksum itself is zero.

### 3.2 Hopper Setup and Initialisation commands

Before the hopper can be used, a number of initialisation steps have to be done.

#### Check communication

Transmit the SIMPLE POLL command to the hopper to check if it is responding with an ACK message:

#### Simple Poll (header 254)

Command : 03 00 01 FE FE  
 Response: 01 00 03 00 FC

Here a normal ACK is received. Device at address 3 is communicating ok.  
 If no response is received then check the hopper address by sending the ADDRESS POLL command:

#### Address Poll (header 253)

Command : 00 00 01 FD 02  
 Response: 03

From the response can be seen that there is only 1 device on the bus with address 3.  
 This command returns all device addresses of the devices on the cctalk bus. If each device has an unique address, then no communication clashes will occur. If a clash occurs, then some devices share the same address. (Address clashes should not occur after power up, since each device will revert to it's default address, which should be unique for every device used. See Table 2: Address select pins). Use the ADDRESS CLASH command (header 252) to check which address clashes. If an address clash occurs, the devices that clashes can be given a random address by using the ADDRESS RANDOM (header 250) command. Send the ADDRESS CLASH again to resolve any other address clashes. Once every device has an unique address, the addresses can be optionally changed to new addresses using the ADDRESS CHANGE (header 251) command.

Once communication is ok, device details can be requested. For your reference all commands are listed below. Most of the commands are not mandatory to operate the hopper.

### **Request comms revision (header 004)**

Command : 03 00 01 04 F8  
Response: 01 03 03 00 31 33 32 63

The 3 bytes (31 33 32) in the response have the following meaning:  
[cctalk level] [major revision] [minor revision]. In our example: 1 3 2.

### **Request comms status variables (header 002)**

Command : 03 00 01 02 FA  
Response: 01 03 03 00 00 00 00 F9

The 3 data bytes (00 00 00) in the response have the following meaning:  
[rx timeouts] [rx bytes ignored] [rx bad checksums]  
This data can be used to test the quality and load of a cctalk network.

### **Clear comms variables (header 003)**

Command : 03 00 01 03 F9  
Response: 01 00 03 00 FC

This command is used to reset the comms status variables to 0.

### **Reset device (header 001)**

Command : 03 00 01 01 FB  
Response: 01 00 03 00 FC

This command resets (software reset) the hopper, after the transmission of the response message. During software resetting (about 50ms) the hopper will not respond to commands. After the reset, all hopper variables (including status flags) will be reset to their default values.  
⇒ A power-up reset (hardware reset) takes about 500 ms, during which the hopper will not respond to any commands.

### **Request Manufacturer id (header 246)**

Command : 03 00 01 F6 06  
Response: 01 0D 03 00 53 75 7A 6F 20 49 6E 74 20 28 4E 4C 29 E8

The data bytes in the response (53 75 7A 6F 20 49 6E 74 20 28 4E 4C 29) give the manufacturer: "Suzo Int (NL)". Remember that the header byte in the response, here 00, means that an ACK is received.

### **Request Equipment Category id (header 245)**

Command : 03 00 01 F5 07  
Response: 01 06 03 00 50 61 79 6F 75 74 74

The data bytes in the response (50 61 79 6F 75 74) stand for "Payout".

**Request Product Code (header 244)**

Command : 03 00 01 F4 08  
 Response: 01 04 03 00 53 43 48 32 E8

The data bytes in the response (53 43 48 32) mean "SCH2", meaning **Suzo Cube Hopper MK2**.  
 The following hopper configurations can be obtained from factory:

Product Code	Description
SCH2	8 encrypted security bytes must be sent with the dispense command
SCH2-NOENCRYPT	8 dummy bytes must be sent with the dispense command
SCH2-USE_SERNR	The hopper serial number must be sent with the dispense command

**Table 4: Hopper Product Codes**

Always check the Product Code first to determine the hopper operation mode. See also 3.3 Hopper Dispense coins procedure.

**Request Serial Nr (header 242)**

Command : 03 00 01 F2 0A  
 Response: 01 03 03 00 4E 46 05 60

The 3 data bytes are the serial number (hex): 4E 46 05. This is decimal 345678.  
 The least significant bytes are transmitted first.  
 If the hopper Product Code is SCH2-USE\_SERNR, then be sure to send this serial number along with the dispense command in order to start a payout. See also Dispense Hopper coins (header 167).

**Request Software Revision (header 241)**

Command : 03 00 01 F1  
 Response: 0B 01 09 03 00 43 75 62 65 20 56 31 2E 31 6E

The response data byte (43 75 62 65 20 56 31 2E 31) mean "Cube V1.1".

**Request data storage availability (header 216)**

Command : 03 00 01 D8 24  
 Response: 01 05 03 00 02 04 08 03 08 DE

The 5 data bytes in the response have the following meaning:  
 [memory type] [read blocks] [read bytes per block] [write blocks] [write bytes per block]  
 In our example the memory type (02) is EEPROM (100.000 write cycles guaranteed),  
 There are 4 readable blocks (0 .. 3) consisting of 8 bytes.  
 There are 3 writeable blocks (0 .. 2) consisting of 8 bytes.  
 Refer to Table 12: EEPROM Memory Description for a description of the data blocks.

**Request buid code (header 192)**

Command : 03 00 01 C0 3C  
 Response: 01 08 03 00 4C 65 76 20 20 20 4C 6F B2

This command returns the hopper option.  
 There is only one option on the cube hopper: Low level detection plates, and these are always fitted.  
 The data bytes in the response mean: " Lev Lo". The "Lev LoHi" response will never appear, since the cube hopper has no means of checking if the high level sensor is mounted or not.

**Request address mode (header 169)**

Command : 03 00 01 A9 53  
 Response: 01 01 03 00 4A B1

This command is used to determine how the hopper address is handled. See next table.

Bit nr	Description
B0	Address is stored in ROM
B1	Address is stored in RAM
B2	Address is stored in EEPROM or battery-backed RAM
B3	Address selection via interface connector
B4	Address selection via PCB links
B5	Address selection via switch
B6	Address may be changed with serial commands (volatile)
B7	Address may be changed with serial commands (non-volatile)

**Table 5: Address mode**

In our example 4A is returned, meaning that the address is stored in RAM, with a selection via the interface connector and the address may be changed with serial commands.

**Enter PIN number (header 218)**

Command : 03 04 01 DA 31 32 33 34 54  
 Response: 01 00 03 00 FC

The 4 bytes (31 32 33 34) transmitted along with the command are the PIN numbers.  
 Here PIN number "1234" is transmitted. Correct pin codes are immediately acked.  
 Wrong pin codes are delayed (235 ms) acked, causing a receive timeout at the host.  
 A factory fresh hopper will have the pin code mechanism disabled (check with Test Hopper command).

To activate the PIN mechanism, send an ENTER NEW PIN CODE command:

**Enter New PIN number (header 219)**

Command : 03 04 01 DB 31 32 33 34 53  
 Response: 01 00 03 00 FC

The 4 bytes (31 32 33 34) transmitted along with the command are the new PIN numbers.  
 In our example "1234" is entered.

⇒ Once a PIN code is set, it can not be changed or disabled.

To save the PIN code you can for example scramble it and store it in Block 0 of the EEPROM.

### Modify variable settings (header 165)

Command : 03 04 01 A5 14 00 64 01 DA  
 Response: 01 00 03 00 FC

The 4 bytes (14 00 64 01) transmitted with the command have the following meaning:  
 <current limit, motor stop delay, payout timeout, single coin mode>

In this example the hopper is set in single coin mode.  
 (0 = multi coin mode (default), 1 = single coin mode)  
 In single coin mode, only 1 coin at a time can be dispensed.  
 ⇒ This mode can only be set to multi coin mode again by resetting the hopper.  
 A hopper reset will set all variable settings to their default values.

### Request variable settings (header 247)

Command : 03 00 01 F7 05  
 Response: 01 06 03 00 22 00 1E 29 62 00 2B

The 6 data bytes in the response (22 00 1E 29 62 00) have the following meaning:

Item	Value N		Scaling and Units	Physical value	Default value
	Hex	Dec			
Motor Current Limit (see Table 10: Electrical specifications)	32	50	N / 15.1 Amp	3.3 A	3.3 A
Motor Stop Delay	0	0	N ms	0 ms	0 ms
Payout Timeout	1E	30	N * 0.333 sec	10 sec	10 sec
Peak current measured	27	39	N / 15.1 Amp	2.6 A	
Supply voltage	62	98	0.2 + N * 0.127 V	12.6 V	
Connector address	0	0	N + 3	3	3

**Table 6: Hopper Variable Settings**

#### [Motor current limit]

If the current through the motor is above the threshold level (3.3 A) during 160 ms, the motor will reverse for 250 ms to clear the blocking.

#### [Motor stop delay]

This is the time delay after the last coin is paid out before stopping. This should ensure a clean coin exit.

#### [Payout Timeout]

This is the total time each coin is allowed to leave the hopper, including some reverse time in jam situations. If the hopper is empty, the motor will stop after 10 sec (default value).

#### [Peak current measured]

This is the maximum motor current measured, and gives you an idea about the peak current your power supply must handle. Start and stop currents in the Cube hopper are software controlled and do not have steep slopes (about 0.5 Amp/ms). If the peak current exceeds the Absolute Maximum Current Level (6.3 Amp) after a delay of 2 seconds then the hopper will return a NAK response on a Start Payout command, because the bit B0 in the Hopper Status Reg1 will be set. A Reset command will clear the error.

#### [Supply voltage]

This is the measured supply voltage the hopper runs on.

#### [Connector address]

The address of the hopper after a power up or reset is equal to this address + 3

### 3.3 Hopper Dispense coins procedure

Dispensing coins using a cctalk hopper requires some extra command steps before the actual dispense command can be executed successfully.

#### Test Hopper (header 163)

Command : 03 00 01 A3 59

Response: 01 02 03 00 C0 80 BA

First of all, check if any error flags are set in the status bytes.

This can be checked by sending a hopper TEST command. Two status bytes are returned in response. See Table 7: Hopper Status Register 1 and Table 8: Hopper Status Register 2.

Hopper Status Register 1		
Bit nr	Description	Default value after power up
B0	1 = Absolute maximum current exceeded	0
B1	1 = Payout timeout occurred	0
B2	1 = Motor reversed during last payout to clear a jam	0
B3	1 = Opto fraud attempt, path blocked during idle	0
B4	1 = Opto fraud attempt, short-circuit during idle	0
B5	1 = Opto blocked permanently during payout	0
B6	1 = Power-up detected	1
B7	1 = Payout disabled	1

**Table 7: Hopper Status Register 1**

Hopper Status Register 2		
Bit nr	Description	Default value after power up
B0	1 = Opto fraud attempt, short-circuit during payout	0
B1	1 = Single coin payout mode	0
B2	1 = Checksum A error	0
B3	1 = Checksum B error	0
B4	1 = Checksum C error	0
B5	1 = Checksum D error	0
B6	1 = Power fail during NV Memory write	0
B7	1 = Pin number mechanism enabled	0 or 1

**Table 8: Hopper Status Register 2**

If any error flags are set, solve the problem by inspecting the hopper and issue a RESET command.

If the hopper is build in the machine, inspecting may be difficult. There may be a coin stuck in the coin exit port, due to a heavy jam followed by a reset, or due to a power failure during a payout. The "Opto fraud attempt, path blocked during idle" flag is then set. It can be cleared again with a Reset command but is set again 333ms after the Reset command, due to the opto-test during idle (done 3x per second). If an Opto error flag is set, no payout can be started. However, if a Dispense command is transmitted within 333ms after a Reset command, the opto-flags will be cleared and the hopper may be started again to start a new payout.

In our example the first status byte (C0) indicates that a power up is detected and the hopper is disabled. The second byte (80) tells us that the PIN number mechanism is enabled. To unlock the hopper, a pin code must be send to the hopper:

If no opto-error flags are set, the hopper enable command may be issued.

After a power up or a reset the hopper is disabled. Check the status bytes again. The first data byte contains the status of the hopper. If bit B7 is set, payout is disabled.

After checking the hopper status bytes, check if any residual (uncompleted) payout is pending. This can occur if a previous payout was aborted due to a power failure, a coin jam or hopper ran out of coins. Transmit the REQUEST HOPPER STATUS command to check the status:

#### **Request hopper status (header 166)**

Command : 03 00 01 A6 56

Response: 01 04 03 00 00 00 01 00 F7

The 4 data bytes (00 00 01 00) in the response have the following meaning:

[event counter] [payout coins remaining] [last payout: coins paid] [last payout: coins unpaid]

#### **[event counter]**

After each valid (no communication errors) Dispense coins command, this counter is incremented. Only after a hopper reset it is set to 0. This counter should be checked each time a dispense command is transmitted, to check if the command has been received by the hopper. This should prevent sending too many or too less payout commands resulting in wrong payouts. If the hopper status event counter in incremented, then check the payout results by checking the coin counters.

#### **[payout coins remaining]**

After receiving a hopper dispense command, this counter is set with the number of coins to pay.

Each time a coin is paid, this counter is decremented.

If the payout operation completes successfully or abnormally, this counter will be set to 0.

The Host software should always check this counter if it has become 0. If it has become 0 and the coins unpaid counter is non-zero, then the dispense procedure has been aborted before all coins were dispensed. Check with the Hopper Test command if the hopper has timed-out (due to jams or empty).

#### **[last payout: coins paid]**

After receiving a hopper dispense command, this counter is set to 0.

Each time a coin is paid, this counter is incremented.

If the payout operation completes successfully, this counter will be equal to the number of coins paid since the last payout.

#### **[last payout: coins unpaid]**

This counter holds the number of coins that failed to payout after the hopper aborted the payout operation. Since the [payout coins remaining] counter is set to 0 after abnormal termination, this counter will hold the number of coins unpaid. During a payout, this counter will be set to 0.

CoinsUnpaid is only saved if the power is lost during a payout (abnormal termination).

If the hopper stops due to a payout timeout or emergency stop (normal terminations), then

CoinsUnpaid is cleared. The host machine is responsible for remembering the nr coins unpaid.

⇒ Coins Unpaid will be set to 0, after executing an 'Emergency Stop' command followed by a power cycle. During the execution of this command, the nr of unpaid coins is reported back to the host machine which will store it. Storing this number also in the hopper is not desirable.

### **Enable Hopper (header 164)**

Command : 03 01 01 A4 A5 B2  
Response: 01 00 03 00 FC

Transmit the ENABLE HOPPER command with data byte 1 set to 165 (= A5 Hex) to enable payout. The hopper can be disabled by sending the ENABLE HOPPER command with data byte 1 set to any value other than 165. After transmitting the ENABLE HOPPER command the TEST HOPPER command can be issued again to check if the payout is enabled.

⇒ The hopper remains enabled until the hopper is reset or is disabled by command.

Once the hopper is enabled (and not blocked by a pin code) the payout can be started.

- If the hopper Product Code is "SCH2", then the DISPENSE COINS command needs an 8-byte security code from the hopper in order to start a payout. Get the security code from the hopper by transmitting the REQUEST CIPHER KEY command. Optionally the PUMP RNG command may be transmitted prior to the Request Cipher Key command to randomize the security code from the hopper even more.

- If the hopper Product Code is "SCH2-NOENCRYPT", then the DISPENSE COINS command still needs an 8-byte code, but the value of the code does not matter.

- If the hopper Product Code is "SCH2-USE\_SERNR", then the DISPENSE COINS command needs it's 3-byte serial number sent along with the command to enable a payout.

### **Pump RNG (header 161)**

Command : 03 08 01 A1 5B DA AA 6F 9A 5D C5 06 43  
Response: 01 00 03 00 FC

The 8 bytes transmitted with the Pump RNG command are random numbers generated by the host. These random numbers are used by the hopper to generate a random cipher code.

### **Request Cipher Key (header 160)**

Command : 03 00 01 A0 5C  
Response: 01 08 03 00 69 EE 8F 1C 25 FA AB 08 20

The Request Cipher Key returns 8 security bytes: (69 EE 8F 1C 25 FA AB 08).

Once the security key from the hopper is received, the payout can be started by transmitting the DISPENSE HOPPER COINS command together with the (encrypted) security bytes and the number of coins to payout. The code the encrypt the cipher keys are in the following document: CMF1-1 from MoneyControls.

⇒ It is possible to disable the security bytes and replace the security bytes with the 3-byte serial number of the hopper. The hopper can be set in this mode using a special Setup program from the factory. Refer to the Product Code to check your version of the hopper. See also Request Product Code (header 244). Remember that when the the unencrypted serial number is used as a dispense key, the Request Cipher Key must also be requested before the dispense command can be issued.

### Dispense Hopper coins (header 167)

Command : 03 09 01 A7 28 DB 6A 16 7C 92 B9 C6 03 39  
Response: 01 01 03 00 02 F9

The Cipher Key from the previous command is sent encrypted (bytes: 28 DB 6A 16 7C 92 B9 C6), together with the dispense coin command (A7) and the nr coins to pay (03).

The response is a ACK message with 1 data byte: [event counter].

Each time a dispense command is transmitted without any communication errors, the event counter is incremented. The host software can check this value if any dispense commands are missing (due to communication errors).

If the hopper Product Code is "SCH2-USE\_SERNR", the dispense command would look like follows:

Command : 03 04 01 A7 DF D7 0A 01 90  
Response: 01 01 03 00 01 FA

The 3-byte serial nr (DF D7 0A) in the command string is followed by the nr of coins to pay (01).

The format of the serial number is explained in Request Serial Nr (header 242).

A NAK response is returned in the following situations (check with HOPPER TEST command):

- The coin exit is blocked
- The hopper is not enabled
- PIN code not transmitted
- Cipher key not requested
- Nr coins is not 1 in single coin mode
- Absolute Maximum Current Level has been exceeded.
- A Hopper Dispense command was sent when the hopper was already dispensing.

When the payout is started, all hopper status counters are updated:

- [event counter] is incremented (also when a NAK is received)
- [payout coins remaining] is set to nr coins to pay
- [coins paid] is set to 0
- [coins unpaid] is set to 0

During payout the the counters are updated as follows:

- [payout coins remaining] is decremented each time a coin is paid
- [coins paid] is incremented a coin is paid
- [coins unpaid] is set to 0

After the payout operation has stopped normally or abnormally, the counters are updated as follows:

- [coins unpaid] is set to [payout coins remaining], except if a Emergency stop occurred due to power being lost. In this case coins unpaid is set to 0, and the host machine should store the number of coins still unpaid (which is returned by the Emergency stop command).
- [payout coins remaining] is set to 0.

**This counter should always be checked during the coin dispensing. If it reached 0, the host software should check if the payment is completed or aborted by checking the coins paid and coins unpaid counters.**

If a power reset occurred during a payout, all counters are saved in EEPROM. These values can be used to finish the pending payout if the power is back again.

During a payout, the coin counters can be retrieved using the REQUEST HOPPER STATUS command. **It is recommended to poll the hopper status each 200ms during payout.**

If the host receives no reply to the REQUEST HOPPER STATUS command, it will be retransmitted 50ms later again.

### Request Hopper Status (header 166)

Command : 03 00 01 A6 56  
 Response: 01 04 03 00 02 00 00 03 F3

The 4 bytes in the response (02 00 00 03) have the following meaning:  
 [event counter] [payout coins remaining] [last payout: coins paid] [last payout: coins unpaid]  
 In our example 2 dispense commands have been issued, 0 coins are remaining, 0 coins are paid and 3 coins are unpaid.

A final step in the payout procedure is:

### Verify payout

After the dispensing of coins, the TEST command should be issued to verify if no abnormal situations have occurred. (opto blocks, jams, etc). The REQUEST HOPPER STATUS and REQUEST HOPPER DISPENSE COUNT commands should be issued to check if all coins have been dispensed properly.

- Check the dispense event counter to check if the dispense command is received.
- Check if any errors occurred during the payout with the hopper test command.
- Check if all coin counters balance with the request hopper status command.
- Check the coin level status (empty/full) (see next command)

### Request Payout high/low status (header 217)

Command : 03 00 01 D9 23  
 Response: 01 01 03 00 10 EB

The returned data byte is the levelstatus.  
 The bits of levelstatus have the following meaning:

Bit nr	Description
0	Low Level status (1 = low level)
1	High Level status (1 = high level)
2	not used
3	not used
4	Low Level sensor is fitted
5	High Level sensor is fitted
6	not used
7	not used

**Table 9: LevelStatus bit definition**

In our example the low level sensor is fitted and the level status is normal.

⇒ The High Level Status can be obtained by connecting the hopper high level sense plates to pin 9 of the hopper connector. If the voltage on pin 9 is 0 volt (shorted to ground through the coins), then the High Level status bit will be set.

The High Level sensor is fitted bit will however always be low, and cannot be used to check if the high level sensor is present.

### **Request Hopper dispense count (header 168)**

Command : 03 00 01 A8 54  
Response: 01 03 03 00 2E 02 00 C9

The 3 data bytes (2E 02 00) represent the 3 byte total dispense counter: 558 (LSB first). This counter is incremented each time a coin is paid, and is not reset before any new payout. The counter can only be reset by writing 4 zero's to the hopper dispense counter which resides in block 2 in EEPROM.

The Hopper dispense life counter is the same counter has the hopper dispense counter, and resides in block 3 in EEPROM. However this block can't be written to and thus the counter can't be reset to 0.

All non-volatile coin counters as well as other EEPROM data can be retrieved with the Read data block command: Refer to Table 12: EEPROM Memory Description.

### **Read Data Block (header 215)**

Command : 03 01 01 D7 02 22  
Response: 01 08 03 00 2E 02 00 D0 01 FF 00 00 F4

In this example Read Data Block 2 is requested.

The 8 bytes in the response (2E 02 00 D0 01 FF 00 00) contain the hopper dispense counter (2E 02 00) and a checksum (D0) over the counter.

### **Write Data Block (header 214)**

Command : 03 09 01 D6 02 00 00 00 00 00 00 00 00 00 1B  
Response: 01 00 03 00 FC

In this example 8 bytes (00 00 00 00 00 00 00 00) were send to EEPROM Block 2. This means that Hopper Dispense count, coins paid and coins unpaid are all reset to 0.

### Summary dispense coins example:

1. Check the hopper status by sending the TEST command:

Command : 03 00 01 A3 59

Response: 01 02 03 00 80 80 FA

The hopper is disabled and the pin code mechanism is enabled.

2. Enable the hopper by sending the ENABLE command:

Command : 03 01 01 A4 A5 B2

Response: 01 00 03 00 FC

Optionally a new TEST command can be issued to verify the status of the hopper.

### Next 2 steps (3 and 4) are not necessary for the cctalk implementation when the Serial Nr is used with the Dispense command. (Hopper Product Code is "SCH2-USE\_SERNR")

3. This step is optional: Randomize hopper security code by sending PUMP RNG command:

Command : 03 08 01 A1 5B DA AA 6F 9A 5D C5 06 43

Response: 01 00 03 00 FC

The security code in the hopper is randomized using 8 random bytes.

4. This step is optional: Request the security bytes from the hopper by sending the REQUEST CIPHER KEY command:

Command : 03 00 01 A0 5C

Response: 01 08 03 00 B9 FE 5F AC 75 0A 7B 98 A0

The following security bytes are received from the hopper: B9 FE 5F AC 75 0A 7B 98

5. If the hopper Product Code is "SCH2", then start the payout of 3 coins by sending the DISPENSE COINS command together with the security codes:

Command : 03 09 01 A7 63 D9 2A 95 BA D4 35 82 03 09

Response: 01 01 03 00 01 FA

The host sends the (encrypted) security code 63 D9 2A 95 BA D4 35 82 together with the number of coins to pay (03).

If the hopper Product Code is "SCH2-USE\_SERNR", the dispense command would look like follows:

Command : 03 04 01 A7 DF D7 0A 01 90

Response: 01 01 03 00 01 FA

The 3-byte serial nr (DF D7 0A) in the command string is followed by the nr of coins to pay (01).

The format of the serial number is explained in Request Serial Nr (header 242).

The data byte in the response is an event counter. Each time the DISPENSE COINS command is issued, the event counter is incremented.

6. During the payout, the number of remaining coins can be checked with the REQUEST HOPPER STATUS command:

Command : 03 00 01 A6 56

Response: 01 04 03 00 02 00 03 00 F3

From the response can be seen that 2 dispense commands have been sent, 0 coins are remaining, 3 coins have been paid and 0 coins are unpaid. Check the Hopper Status each 200ms during payout. Once [Nr Coins Remaining] is 0, the polling may be stopped and the payout can be verified.

7. Check the hopper status by sending the TEST command:

Command : 03 00 01 A3 59

Response: 01 02 03 00 00 80 7A

From the status bytes can be seen that no payout timeout, opto blockings or coin jams have occurred.

8. Optionally the payout can be verified by checking the coins paid, coins unpaid and total dispense counters using the REQUEST HOPPER STATUS (header 166) and REQUEST HOPPER DISPENSE COUNT (header 168) commands.

### Host program example (only used for protocol explanation) in pseudo code

```

/* Initialize communication */
For each Hopper with HOPPER_ADDRESSx in machine do
{
  if ( SimplePoll(HOPPER_ADDRESSx) ) != ACK)
  {
    // resolve any device address problems
    While ( AddressClash(HOPPER_ADDRESSx) ) // more devices share HOPPER_ADDRESSx ?
    {
      RandomizeAddresses(0); // give each device a new random address
      DeviceList = AddressPoll(0); // store all received address in list

      While (DeviceList not empty)
      {
        address = GetNextAddressFromDeviceList();
        RequestEquipmentCategory(address); // check device type
        ChangeAddress(address, DEVICE_ADDRESS); // change address to DEVICE_ADDRESS
      }
    }
    // SimplePoll(HOPPER_ADDRESSx) returned ACK. Communication Ok!

    if (RequestEquipmentCategory(HOPPER_ADDRESSx) != "Payout")
    {
      ShowMessage("DEVICE IS NOT A HOPPER!");
    }
    else
    {
      Hopper[x].PhysicalAddress = RequestVariableSettings(PHYSICAL_ADDR,
                                                         HOPPER_ADDRESSx);
      Hopper[x].SerialNr = RequestVariableSettings(SER_NR, HOPPER_ADDRESSx);
      // Optional the following items may be retrieved
      Hopper[x].ManufacturerID = ...
      Hopper[x].ProdCode = ...
      Hopper[x].SoftwareRev = ...
      Hopper[x].CommsRev = ...
      Hopper[x].HopperCoin = ...
      Hopper[x].BuildCode = ...
    }
  }
}

// Product Configuration details

// Check for pending Payout after a power fail recovery

```

```

// Dispense Coins from HOPPER_ADDRESSx
HopperAddress = HOPPER_ADDRESSx;
HopperStatus = TestHopper(HopperAddress);           // get hopper status bytes
HopperType = RequestProductCode(HopperAddress);     // get hopper type

// Pin Number Unlocking
if (HopperStatus & PIN_NUMBER_REQUIRED)
{
    SetPinNumber(PinCode, HopperAddress);           // unlock hopper
}

// Enable Hopper
if (HopperStatus & HOPPER_DISABLED)
{
    EnableHopper(HopperAddress);                   // enable hopper
}

// Start payout
if (HopperStatus & PAYOUT_ERROR_FLAGS) == 0) // no opto-errors, max current exceeded, etc?
{
    SecurityBytes = RequestCipherKey(HopperAddress); // ReqCipherKey, for all hopper types

    if (HopperType == "SCH2" ) // use security bytes?
    {
        DispenseCoins(HopperAddress, SecurityBytes, NrCoinsToPay);
    }
    else if (HopperType == "SCH2-NOENCRYPT") // use dummy bytes? (security disabled)
    {
        DispenseCoins(HopperAddress, DummyBytes, NrCoinsToPay);
    }
    else if (HopperType == "SCH2-USE_SERNR") // use serial nr as dispense key?
    {
        SerialNr = RequestSerialNumber(HopperAddress);
        DispenseCoins(HopperAddress, SerialNr, NrCoinsToPay);
    }
}

// Check hopper status during payout
HopperCounters = RequestHopperStatus(HopperAddress); // HopperCounters is a structure
while (HopperCounters.NrCoinsRemaining) > 0)
{
    Delay(200ms); // implementation may of coarse be interrupt driven (timer)
    HopperCounters = RequestHopperStatus(HopperAddress); // update status counters
}

// Verify Dispense procedure (may be extended of coarse)
HopperStatus = TestHopper(HopperAddress); // get hopper status bytes
if (HopperCounters.NrUnPaidCoins > 0) // payout completed ?
{
    if (HopperStatus & PAYOUT_TIMEOUT_OCCURED) // no, hopper timeout ?
    {
        ShowMessage("Hopper Timeout occurred");
    }

    if (HopperStatus & JAM_OCCURED) // hopper jammed during payout ?
    {
        ShowMessage("Hopper probably jammed");
    }
}

// Check Hopper Levels
HopperLevel = RequestHopperCoinLevel(HopperAddress);
if (HopperLevel & LOW_LEVEL)
{
    ShowMessage("Hopper nearly empty");
}

if (HopperLevel & HIGH_LEVEL)
{
    ShowMessage("Hopper nearly full");
}

// Check Total Dispense counter, etc

```

### 3.4 Coin Jams during payout

During a payout, a coin may block the hopper (for example if the hopper is loaded too heavily, or a wrong coin has slipped into the hopper). In this case the motor current will rise quickly. A fully blocked hopper motor will draw peak currents up to 7 Amp. When the power supply is not able to deliver this peak current, the voltage on the hopper will drop.

We recommend the following power supply types from Suzo:

- 42PP0520 (dual supply: 5VDC and 12VDC)
- 42PP0530 (dual supply: 5VDC and 24VDC)

When the voltage drops below the POWER\_FAIL\_TRESHOLD (see Table 10: Electrical specifications) during 20ms, the hopper will reset, aborting the payout procedure. This 20ms may seem a short time, but remember that if the power supply is 8.0V and falling, the hopper must be able to stop and save all payout settings before the power is gone totally.

As long as the power supply on the hopper is above the POWER\_FAIL\_TRESHOLD level, the hopper will start an anti-jam operation by reversing and restarting the hopper motor.

If the motor current is greater than 3.3A during 160ms, the motor will start reversing in order to un-jam the hopper. During anti-jamming, the current may rise to levels of 6 Amp peak.

### 3.5 Power failures

The hopper measures the voltage each ms.

If the voltage drops below the the POWER\_FAIL\_TRESHOLD during 20ms, the hopper will stop immediately if it was running, and save all counters (CoinsPaid, CoinsUnpaid) in Non-volatile memory.

A power dip of more than 20ms will stop the hopper and save all data.

If the power returns again, the host software must retrieve the hopper status and take appropriate action if there are, for example, unpaid coins left.

If the host machine has early-power down notification, the host machine may send an 'Emergency Stop' command to the hopper so that the hopper will stop and save all data.

The NrCoinsRemaining is transmitted back to the host. The host machine must ensure that this value is stored in its own non-volatile memory.

## 4. Electrical & Timing Specifications

Parameter	Min	Typ @ 12V	Typ @ 24V	Max	Units
Supply voltage	11	12	24	26	Vdc
Idle current		30	30		mA
Motor_start current		2.2	2.2	5	A
Motor_running_current		0.5	0.6	0.9	A
Vtrip (Power fail threshold)		8.0 during 20ms	8.0 during 20ms		Vdc
Motor_current_reverse level		> 3.3 A during 160 ms	> 3.3 A during 160 ms		A
Thermal fuse protection		5.5	5.5		A

**Table 10: Electrical specifications**

Parameter	Description	Value	Units
Brate	Serial communication speed	9600	baud
Trxout	Receive data timeout	25	ms
TPinit	Power up initialisation time	< 600	ms
TRinit	Hardware reset init time	< 20	ms
TSinit	Software reset init time	< 40	ms
PWMFreq	Motor drive PWM frequency	20	kHz
TMreverse	Motor reverse time	250	ms
Ifuse	Fuse trip time @ 5.5 A	5	sec
Tlevdeb	Level sensor debounce time	2	sec
TEESave	EEProm write time per byte	7	ms

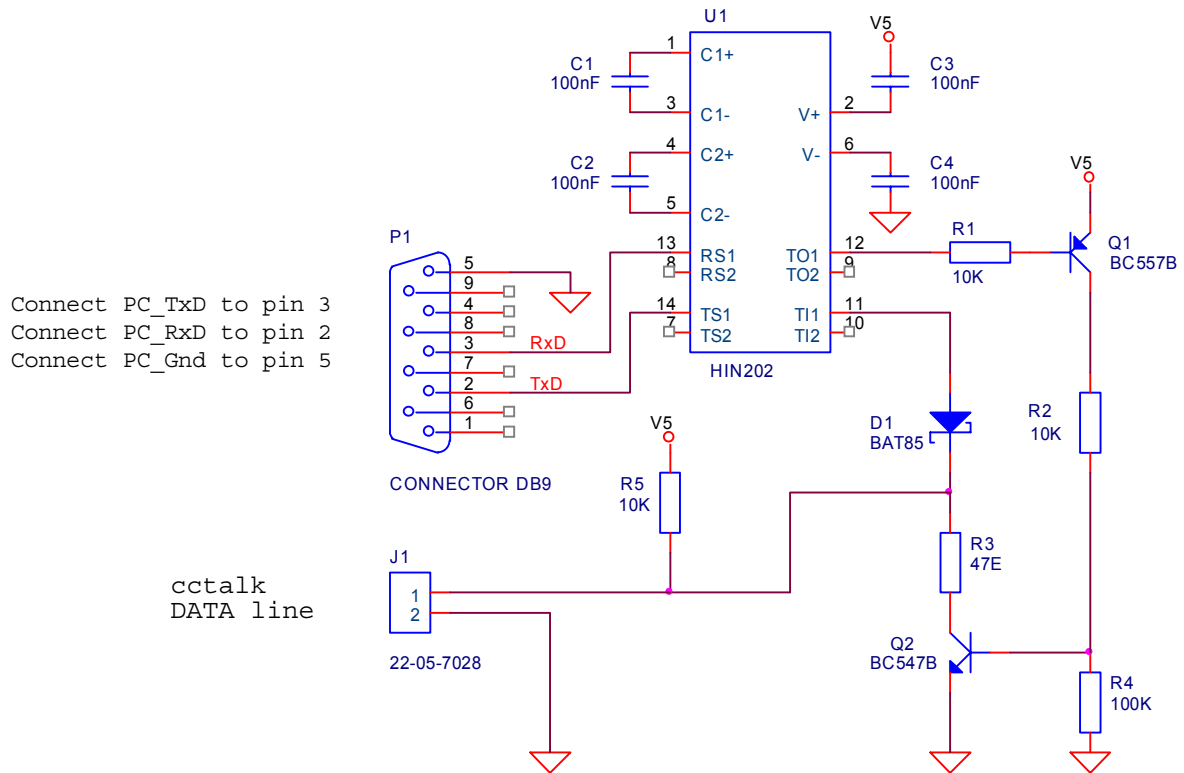
**Table 11: Timing specifications**

Block Nr.	Length (bytes)	Description	Read/Write Permission
0	8	User data	R / W
1	6	Coin name	R / W
1	2	User data	R / W
2	3	Hopper dispense count	R / W
2	1	Checksum A	R / W
2	1	Last payout: coins paid	R / W
2	1	Checksum B	R / W
2	1	Last payout: coins unpaid	R / W
2	1	Checksum C	R / W
3	3	Hopper life dispense count	R
3	1	Checksum D	R
3	1	Black Box Recorder A	R
3	1	Black Box Recorder B	R
3	1	Black Box Recorder C	R
3	1	Black Box Recorder D	R

**Table 12: EEPROM Memory Description**

## Appendix 1. PC Interface Circuit

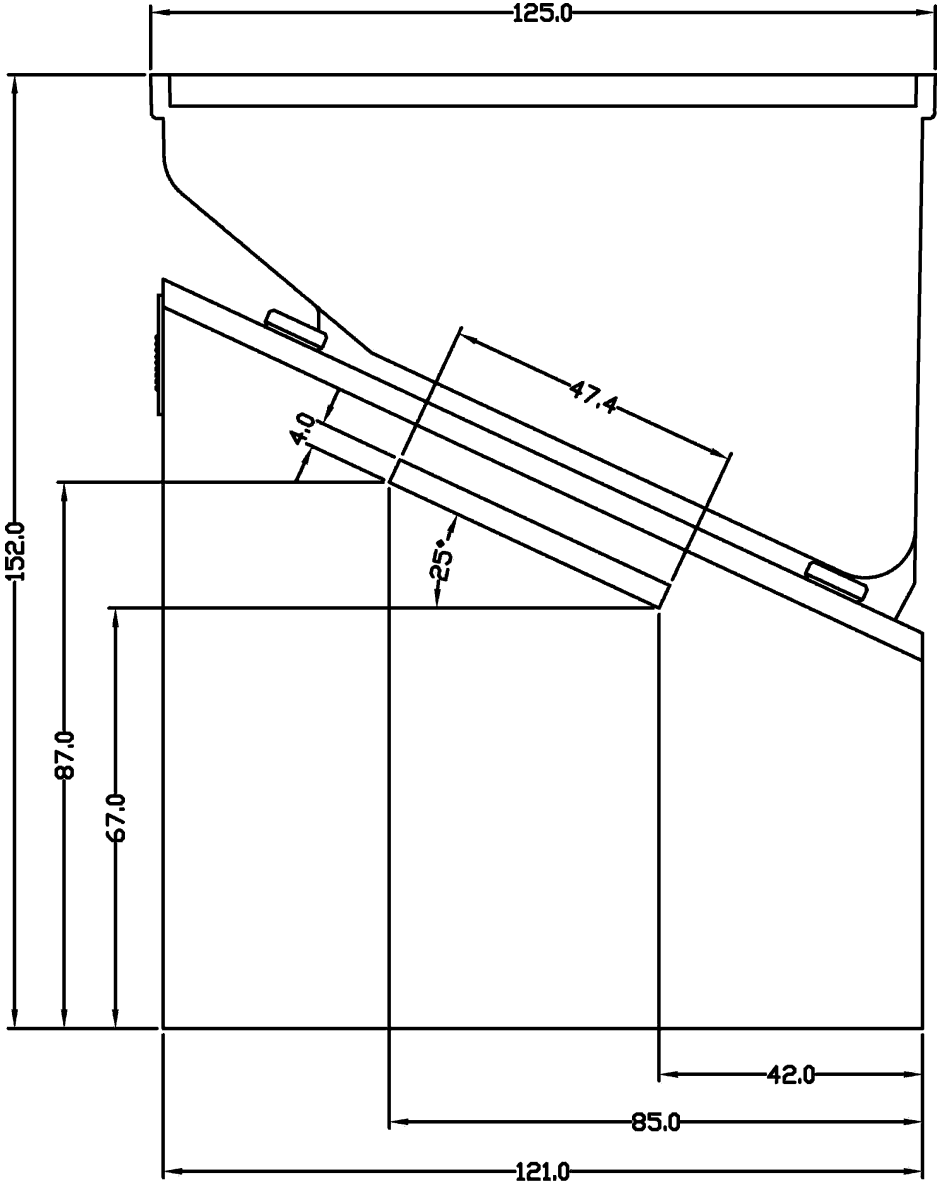
The circuit diagram below shows an interface that can be used to connect a 9-pin serial port of a PC to a cctalk data bus.



### Appendix 2: Mechanical dimensions

Dimensions:  
D: 122mm  
W: 122mm  
H: 153mm

Weight:  
780 grams



PAY-OUT SIDE

## Appendix 3. Maintenance

### Removing the Motor and Gearbox

The motor is controlled by the PCB and operates the disc by means of gears and shafts.

The gearbox is constructed according to the sun and planet system.

The motor can easily be exchanged by first unplugging the connector from the PCB and then dismantling the motor holder which is connected to the gear box by means of a bayonet catch.

Press down the engraved slide on the motor housing and turn the motor housing in an anti-clockwise direction to take it off.

Now take a new motor and refit into the motor holder.

The gearbox can also be removed completely by unscrewing the three screws, which attach the gearbox to the platform and removing the rolling-pin on top of the motor shaft.

The Counter Pawl fits an Opto output detector with a transmitter inside the Hopper. The Counter Pawl makes sure that the coin always leaves the Hopper and the construction is done in such a way that once the coin leaves the disc it can never return to the Hopper.

**Important:** shut-off power from the host machine before starting any cleaning activities.

### Removing and re-installing the hopper

The hopper can easily be removed by pressing on the red release button on the mounting plate and then removing the connector.

Re-installing the hopper:

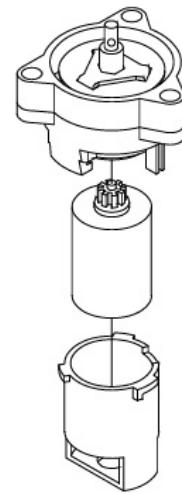
First plug the connector into the Hopper at the reverse side and slide it into the two red hooks on the mounting bracket, then press the red button with lever slightly backwards and push the hopper onto the bracket.

### Cleaning and Materials

All reachable places where the coins pass through the hopper should be cleaned at least every 6 months or after counting 500,000 coins (whichever situation occurs first) with a cloth with dry silicon spray (Part No. 20-0124-1).

In particular the opto coupler needs to be kept clean.

Dirt could obstruct or disrupt the optic signal, resulting in unreliable coin counting.



## **Disassembly**

### **How to remove the disc**

Slide the green button on the back to “down” position.  
Remove the coin cup by sliding it upwards and lift the cup from the platform.  
You are now able to remove the disc from the platform.

### **How to remove the coin insert plate**

Remove the cup and disc from the platform as described above.  
Remove the outlet bridge by pulling it up firmly.  
Gently lift up the coin insert plate using a small screwdriver.

### **How to re-configure the Cube Hopper**

Remove the coin cup.  
Remove the disc and/or coin insert plate (if necessary) and replace them with the ones necessary to achieve the desired configuration.  
Before placing the new disc onto the platform:  
Please check that the correct coin insert plate is mounted on the platform under the bridge. (See Reference Chart)  
Don't forget to put the black Teflon bearing in the centre of the platform. Make sure that the metal ring in the disc is present as well.  
Bring the coin cup onto the platform and slide the green button to the up position to lock it in place.  
NOTE: Please look up the correct configuration for your hopper in the Reference Chart.

## **Appendix 4: TroubleShooting**

### **Coins fail to unjam**

Are you using a solid power supply? (12V, 5A peak)

We recommend the following types from Suzo:

- 42PP0520 (dual supply: 5VDC and 12VDC)
- 42PP0530 (dual supply: 5VDC and 24VDC)

Are you using the correct coin insert plate? (see Reference Chart)

Are you using the right disc? (see Reference Chart)

Be sure Opto Coupler is clear!

Are there bad or incorrect coins in the hopper.

### **Motor fails to run.**

Check the hopper fuse of the host machine.

Protection device tripped - wait for 30 seconds with the power switched off.

### **Over payout of coins.**

Check the opto coupler for accumulated dirt or dust.

Check exit monitoring by the host machine.

Check if the hopper's power is not disconnected too slowly. Power should be disconnected directly after the registration of the ejection of the last coin of the payout.

Are you using the correct coin insert plate? (see Reference Chart)

### **Under payout of coins.**

Make sure the hopper has sufficient coins.

Incorrect registration by the host machine.

Incorrect exit output, debouncing by the host machine.

Bad contact with the hopper.

## **CCTalk communication trouble shooting**

### **No response received after transmitting a command**

- Check if the message contains the proper hopper address.
- Check if the transmitted message is correct (checksum ok. etc).
- Use a PC test program (can be obtained from Suzo) to check if the message is correct.
- Check the quality of the cctalk data line signal using a digital oscilloscope.
- Check if you can see the response from the hopper on the oscilloscope.

## Appendix 5: Optional add-ons

### Cup extension (Part No. 10-0200)

A 65mm high extension together with standard coin-cup increases the capacity of the cup from 400 to 700 coins of €0.50 (24.25mm in diameter, 2.4mm thick). It can be used with all coins with diameters up to 31mm.

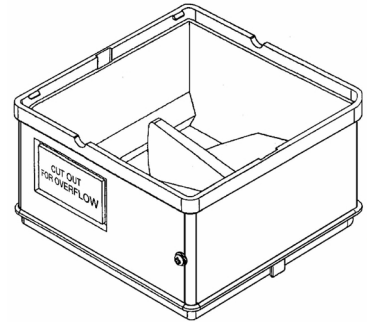
A second extension is possible for coins with a maximum diameter of 25.6mm. This increases the capacity to 1,000 coins of €0.50.

Hopper dimensions - with one extension:      Weight

D: 122mm W: 122mm H: 218mm      860 grams

Hopper dimensions - with two extensions:

D: 122mm W: 122mm H: 284mm      940 grams



## Appendix 6: Payout discs and inserts

<b>Payout discs</b>				
New mark	Old mark	Diameter (mm)	Thickness (mm)	Part no
2	A	18.00 - 22.09	1.50 - 2.09	10-0240-2
22		18.00 - 22.09	1.30 - 1.45	10-0240-22
3	AA	18.00 - 22.09	2.10 - 3.20	10-0240-3
4	B	22.10 - 25.59	1.50 - 2.09	10-0240-4
44		22.10 - 25.59	1.30 - 1.45	10-0240-44
5	BB	22.10 - 25.59	2.10 - 3.20	10-0240-5
6	C	25.60 - 30.09	1.50 - 2.09	10-0240-6
7	CC	25.60 - 30.09	2.10 - 3.20	10-0240-7
8		25.60 - 31.00	1.50 - 2.09	10-0240-8
9		25.60 - 31.00	2.10 - 3.20	10-0240-9
99		€0.20 - €0.50	€1.= - €2	10-0240-99

**Table 13: Payout discs**

<b>Coin insert plates</b>		
Plate mark	Diameter (mm)	Part no
11	18.00 - 18.99	10-0238-11
A	19.00 - 22.09	10-0238
B1	22.10 - 23.89	10-0238-1
B2	23.90 - 25.59	10-0238-2
C1	25.60 - 27.89	10-0238-3
C2	27.90 - 30.09	10-0238-4
5	30.10 - 31.00	10-0238-5
7	for €0.20	10-0238-7

**Table 14: Coin insert plates**

### **PLEASE NOTE:**

Changes have been made to the previously used configuration of the Cube Hopper.

- Payout disc 10-0240-6: replaced by 10-0240-8.
- Payout disc 10-0240-7: replaced by 10-0240-9.
- Hopper 10-1900-60: replaced by 10-1900-83.
- Hopper 10-1900-61: replaced by 10-1900-84.
- Hopper 10-1900-70: replaced by 10-1900-93.
- Hopper 10-1900-71: replaced by 10-1900-94.

### Which hopper to use for which Euro-coin:

<b>€0.01</b>	Not available	<b>€0.20</b>	10-1700-25
<b>€0.02</b>	10-1700-15	<b>€0.50</b>	10-1700-99
<b>€0.05</b>	10-1700-20	<b>€1.00</b>	10-1700-99
<b>€0.10</b>	10-1700-20	<b>€2.00</b>	10-1700-99

## Appendix 7: Reference chart

### Model no. **10-1900-15**

Diameter 18.00 to 18.99mm. Thickness 1.50 to 2.09mm.  
Disc is marked **"2"** (Part no. **10-0240-2**).  
Use coin insert plate **11** (Part no. **10-0238-11**).

### Model no. **10-1900-20**

Diameter 19.00 to 22.09mm. Thickness 1.50 to 2.09mm.  
Disc is marked **"2"** (Part no. **10-0240-2**).  
Use coin insert plate **A** (Part no. **10-0238**).

### Model no. **10-1900-30**

Diameter 19.00 to 22.09mm. Thickness 2.10 to 3.20mm.  
Disc is marked **"3"** (Part no. **10-0240-3**).  
Use coin insert plate **A** (Part no. **10-0238**).

### Model no. **10-1900-40**

Diameter 22.10 to 23.89mm. Thickness 1.50 to 2.09mm.  
Disc is marked **"4"** (Part no. **10-0240-4**).  
Use coin insert plate **B1** (Part no. **10-0238-1**).

### Model no. **10-1900-41**

Diameter 23.90 to 25.59mm. Thickness 1.50 to 2.09mm.  
Disc is marked **"4"** (Part no. **10-0240-4**).  
Use coin insert plate **B2** (Part no. **10-0238-2**).

### Model no. **10-1900-50**

Diameter 22.10 to 23.89mm. Thickness 2.10 to 3.20mm.  
Disc is marked **"5"** (Part no. **10-0240-5**).  
Use coin insert plate **B1** (Part no. **10-0238-1**).

### Model no. **10-1900-51**

Diameter 23.90 to 25.59mm. Thickness 2.10 to 3.20mm.  
Disc is marked **"5"** (Part no. **10-0240-5**).  
Use coin insert plate **B2** (Part no. **10-0238-2**).

### Model no. **10-1900-83**

Diameter 25.60 to 27.89mm. Thickness 1.50 to 2.09mm.  
Disc is marked **"8"** (Part no. **10-0240-8**).  
Use coin insert plate **C1** (Part no. **10-0238-3**).

### Model no. **10-1900-84**

Diameter 27.90 to 30.09mm. Thickness 1.50 to 2.09mm.  
Disc is marked **"8"** (Part no. **10-0240-8**).  
Use coin insert plate **C2** (Part no. **10-0238-4**).

### Model no. **10-1900-85**

Diameter 30.10 to 31.00mm. Thickness 1.50 to 2.09mm.  
Disc is marked **"8"** (Part no. **10-0240-8**).  
Use coin insert plate **5** (Part no. **10-0238-5**).

### Model no. **10-1900-93**

Diameter 25.60 to 27.89mm. Thickness 2.10 to 3.20mm.  
Disc is marked **"9."** (Part no. **10-0240-9**).  
Use coin insert plate **C1** (Part no. **10-0238-3**).

### Model no. **10-1900-94**

Diameter 27.90 to 30.09mm. Thickness 2.10 to 3.20mm.  
Disc is marked **"9."** (Part no. **10-0240-9**).  
Use coin insert plate **C2** (Part no. **10-0238-4**).

### Model no. **10-1900-95**

Diameter 30.10 to 31.00mm. Thickness 2.10 to 3.20mm.  
Disc is marked **"9."** (Part no. **10-0240-9**).  
Use coin insert plate **5** (Part no. **10-0238-5**).

## THE “EURO-HOPPERS”

Model no. **10-1900-25**, "Euro-hopper"

For €0.20 coins.

Disc is marked "**99**". (Part no. **10-0240-99**).

Use coin insert plate **7** (Part no. **10-0238-7**)

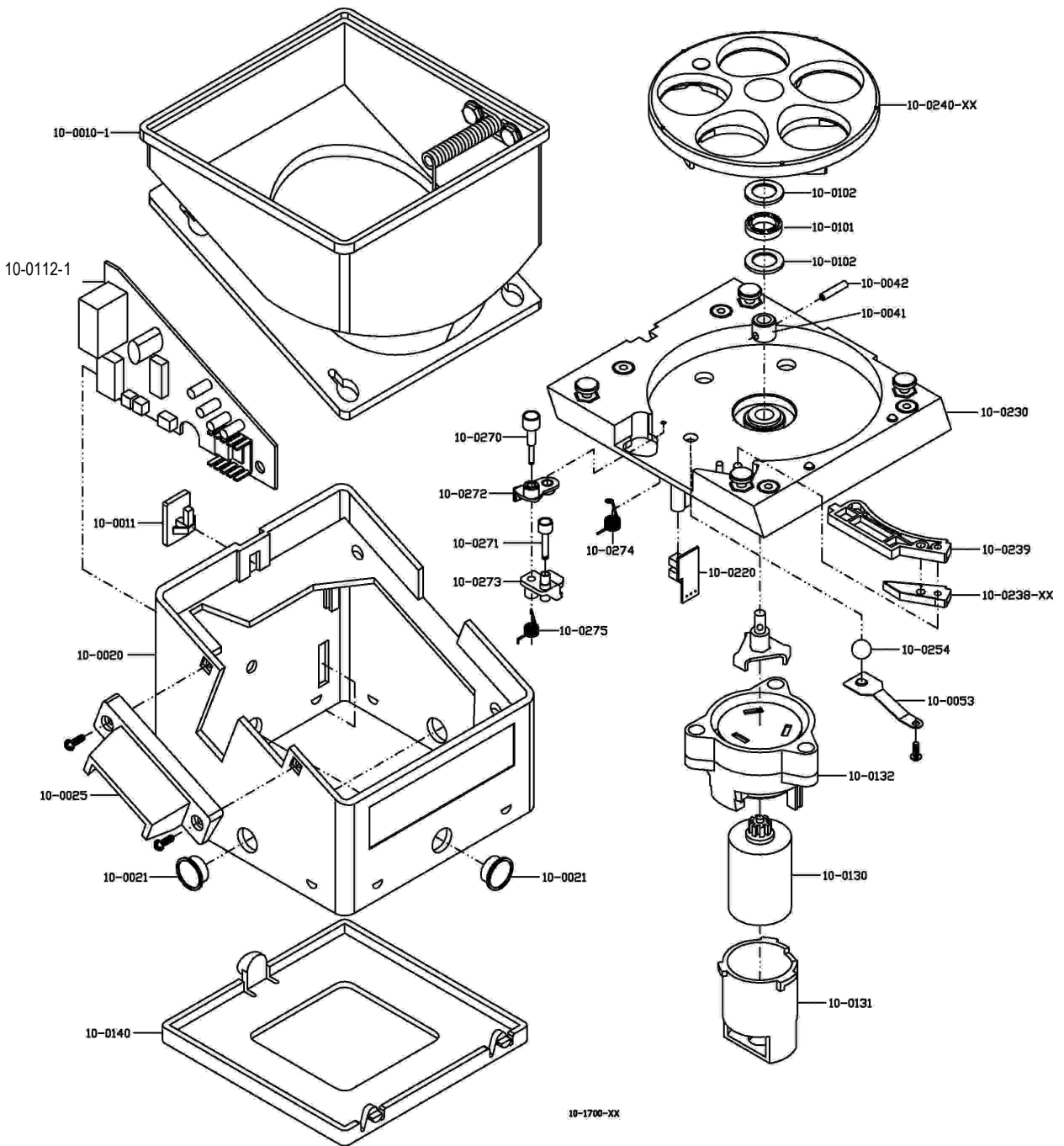
Model no. **10-1900-99**, "Euro-hopper"

For €0.50, €1.00 and €2.00 coins.

Disc is marked "**99**". (Part no. **10-0240-99**).

Use coin insert plate **B1** (Part no. **10-0238-1**)

### Appendix 8: Exploded view



## Appendix 9: CCTalk Cube Hopper Test Program

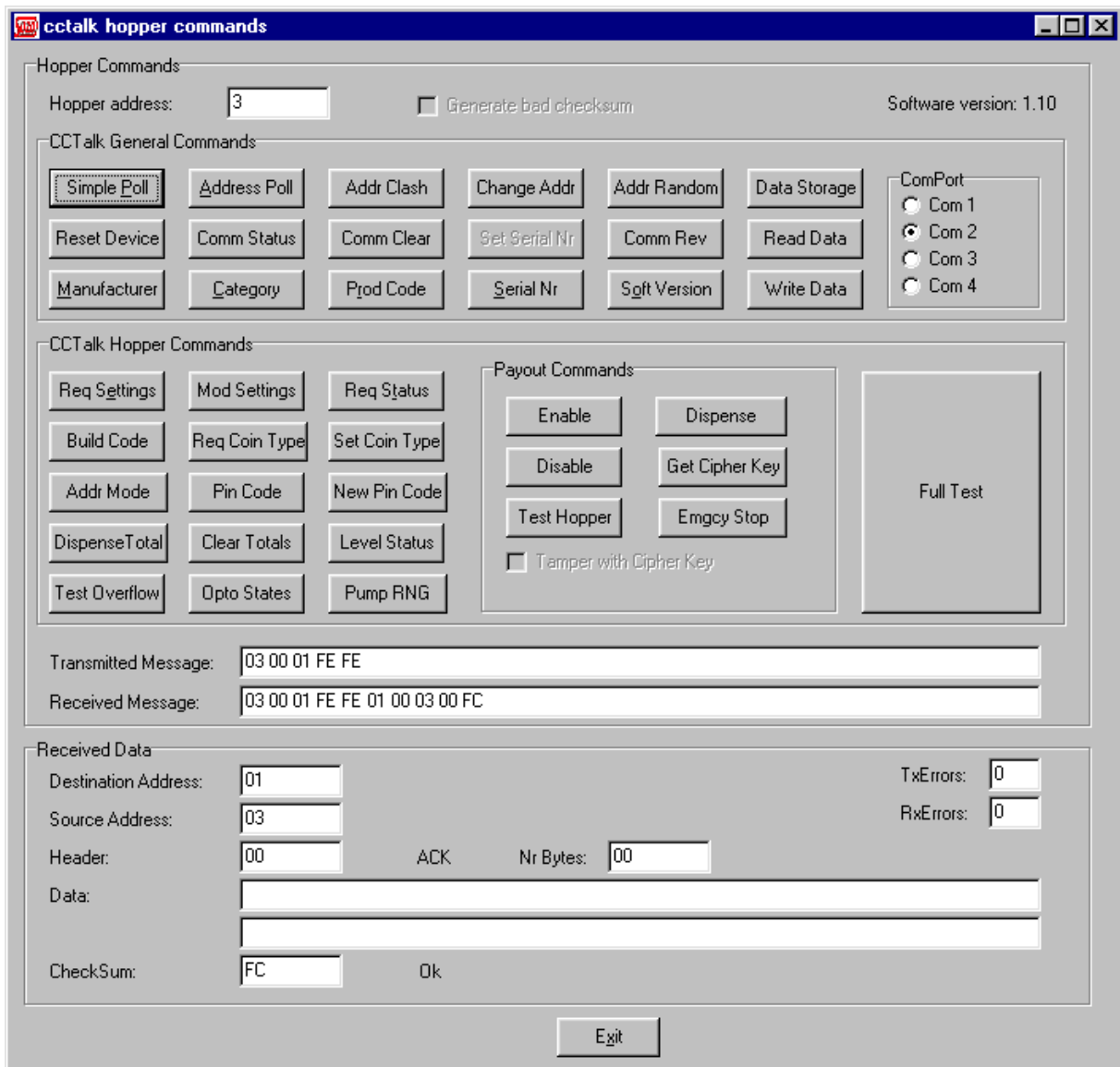
The cctalk implementation on cube hopper can be tested and verified using a test program on the PC. This program can be obtained from Suzo. Refer to the CCTalk Cube Hopper Test Program.

You need to use the PC Interface described in Appendix 1 between the PC and the Cube Hopper in order to connect the PC to the Hopper.

PC-Interface units can be ordered from Suzo (Article nr. 107-0250).

### Installation

- Install the test program on your PC by running the setup program.
- Connect the hopper to the PC using the PC-Interface from appendix 1.
- Start the program. The following screen appears:



- Check if the Simple Poll command returns an ACK.
- If not, check if the transmitted message is echoed back to the PC. If not, check your comport settings and interface circuit.
- Check if the hopper address is ok (default it will be 3).

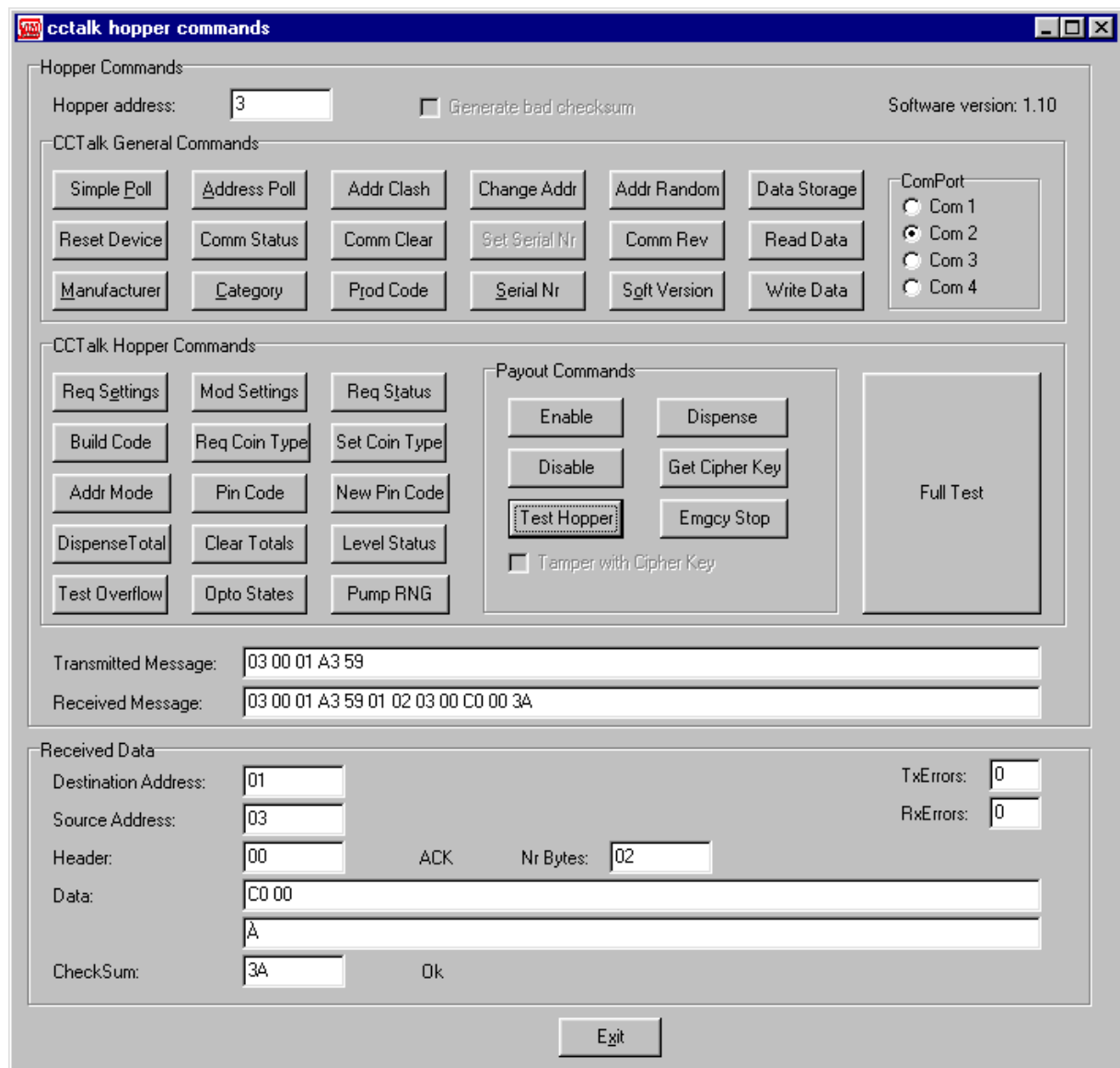
If you have communication (Poll Command is ACK'ed), then all hopper related cctalk commands can be tested.

**Example1: Payout 3 coins**

The default dispense procedure is as follows:

- Retrieve the hopper status (This step is optional, but it gives you information about the hopper status).
- Enable the hopper
- Request serial number if not already done so.
- Transmit Dispense command with a given nr coins.

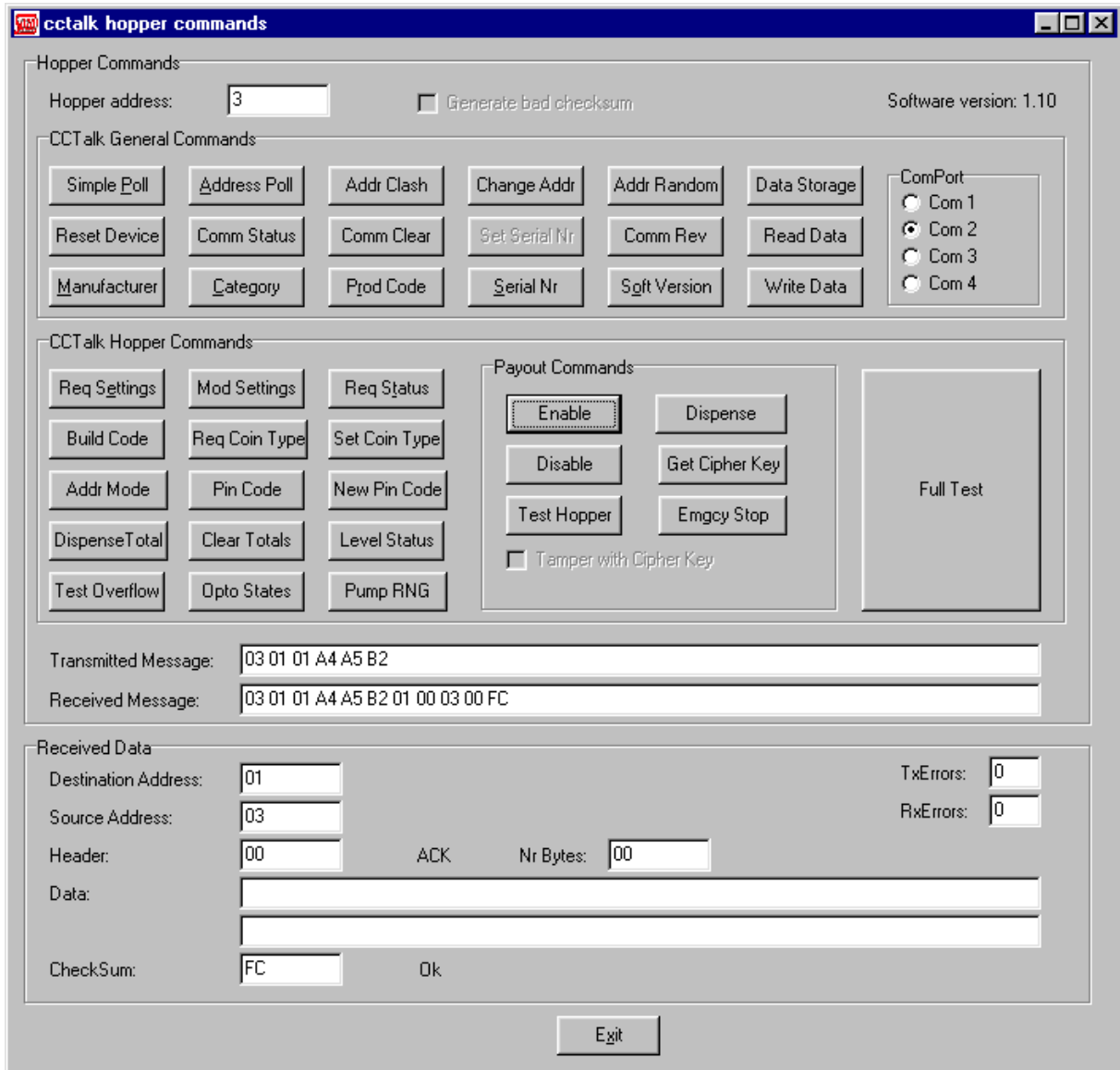
**Step 1: Transmit Hopper Test command**



Two status bytes are returned: C0 00 after initial power up of the hopper. Refer to Table 7: Hopper Status Register 1 and Table 8: Hopper Status Register 2 for the definitions of the status registers. C0 means that a power up is detected and the hopper is disabled.

## Step 2: Enable Hopper

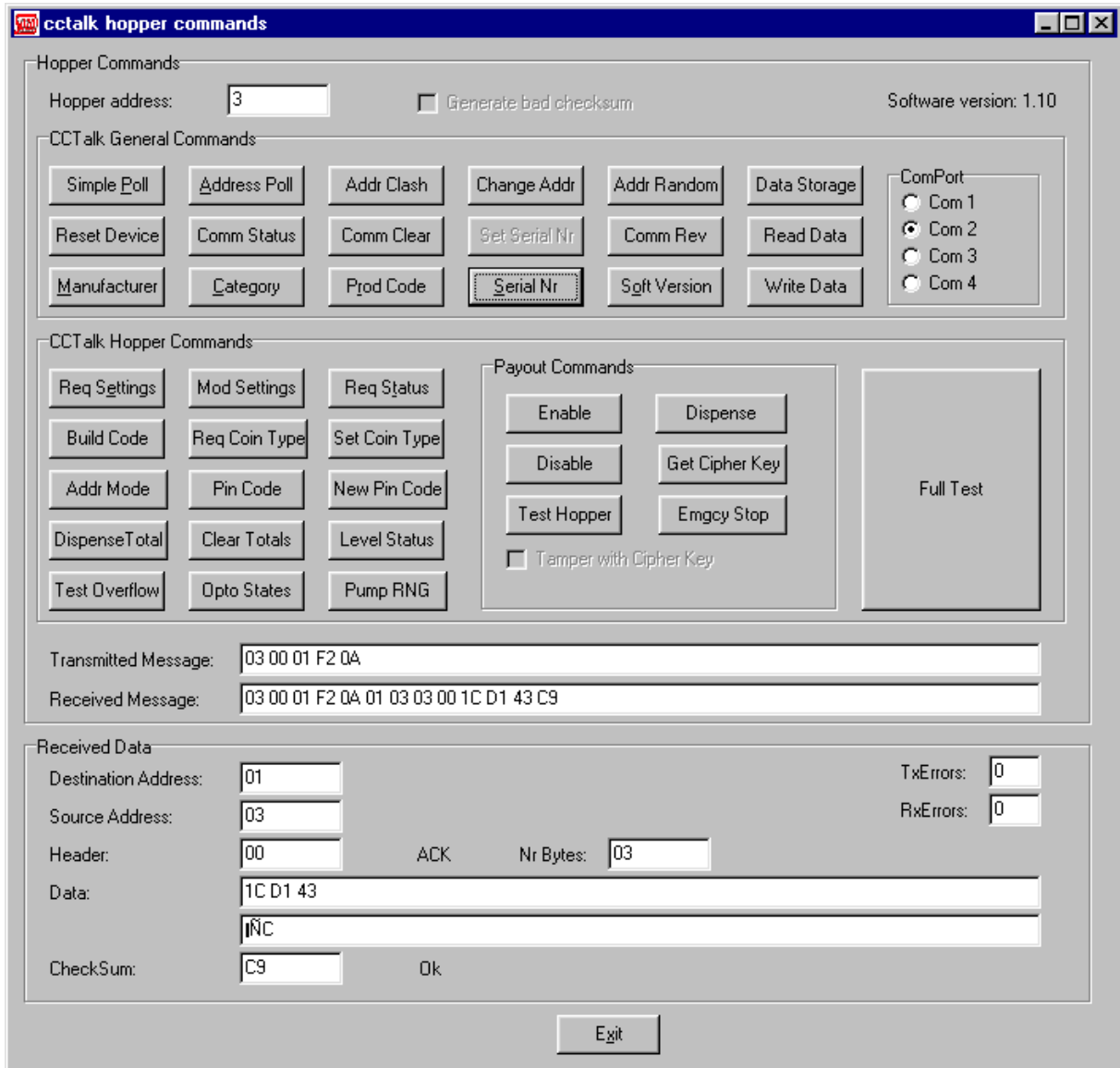
From the Hopper status byte you can see that the hopper is disabled. So we have to enable it first.



The response from the hopper is an ACK (Acknowledge) message.

### Step 3: Get Serial Number from the Hopper

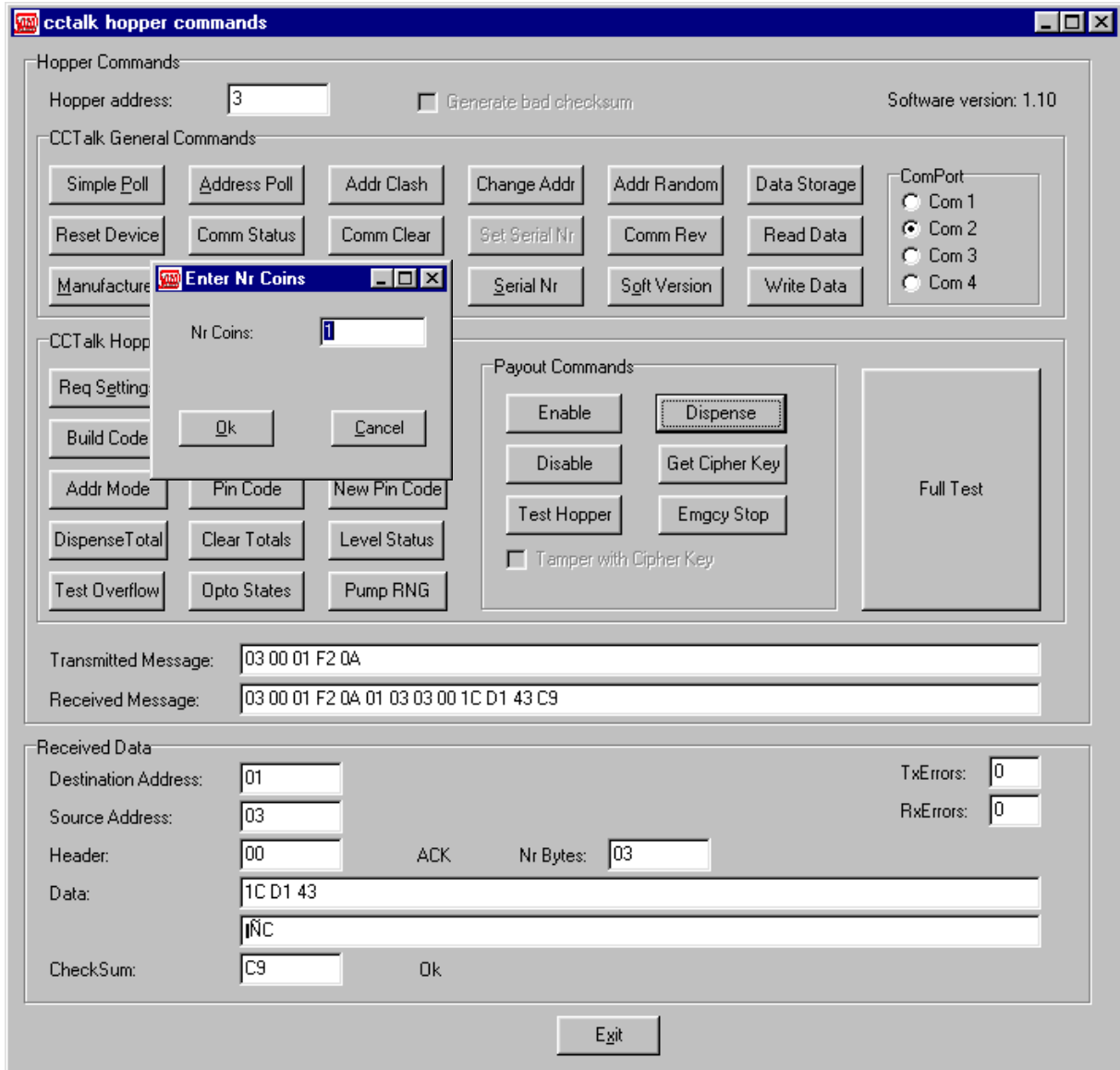
The hopper serial number is needed to start a payout (Non-Encrypted version).



The serial number consists of 3 bytes. Here the serial number is 1C D1 43 (hexadecimal), which is decoded as 4444444 (decimal). This number will be used in the Dispense command.

### Step 4: Transmit Dispense command

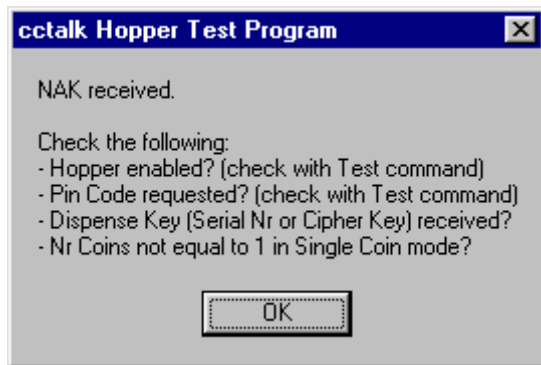
The Dispense command transmits a 3 byte serial number along with the number of coins to pay to the hopper. The Hopper will check if the serial number information in the dispense command matches with it's own serial number. If the match is ok, the hopper will start a payout.



The program asks for a number of coins to payout. Enter a number (between 1 and 255) and press Ok.

The hopper should now start a payout.

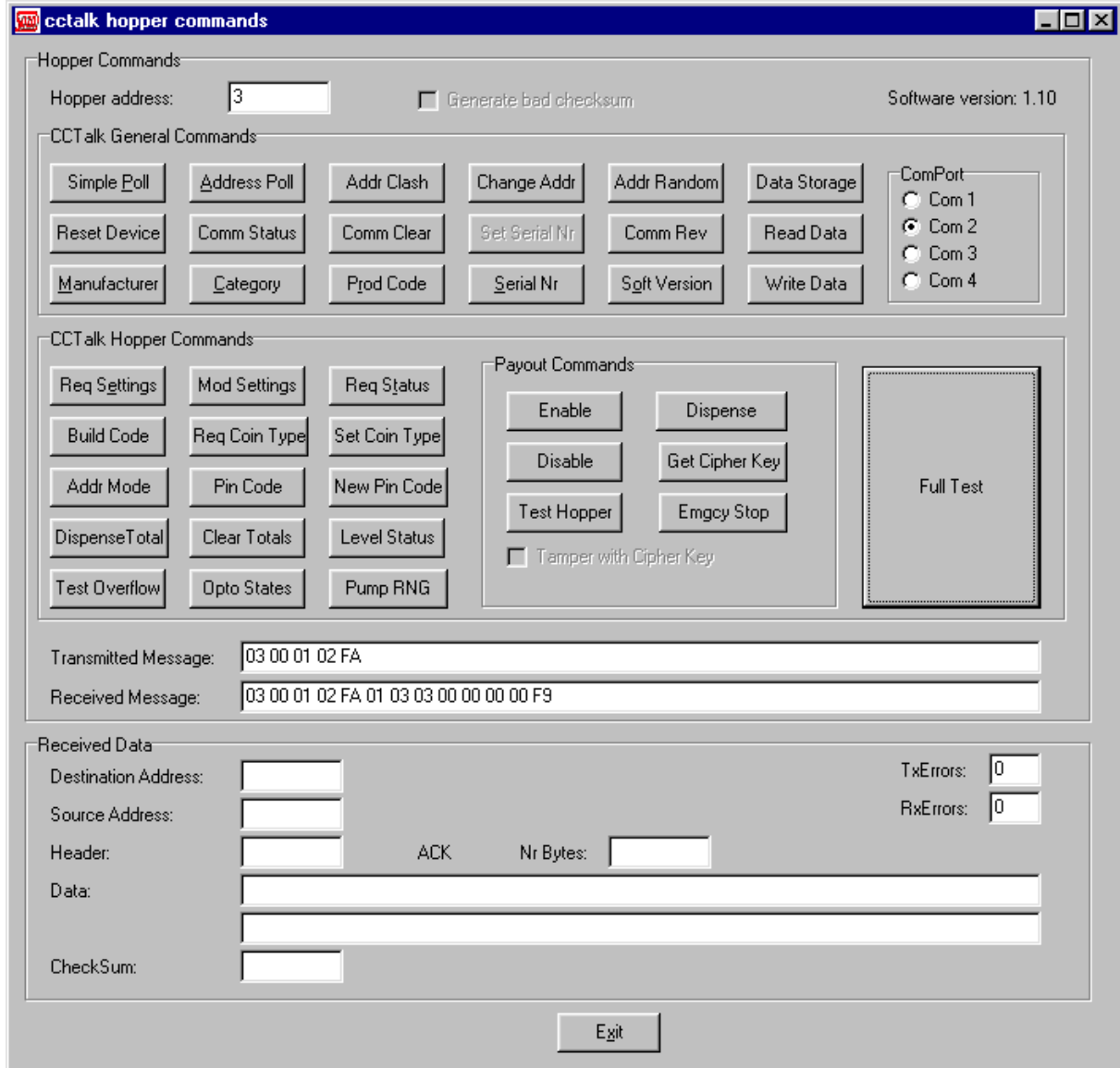
If you skipped one of the necessary steps to start a payout, the next message will be displayed.



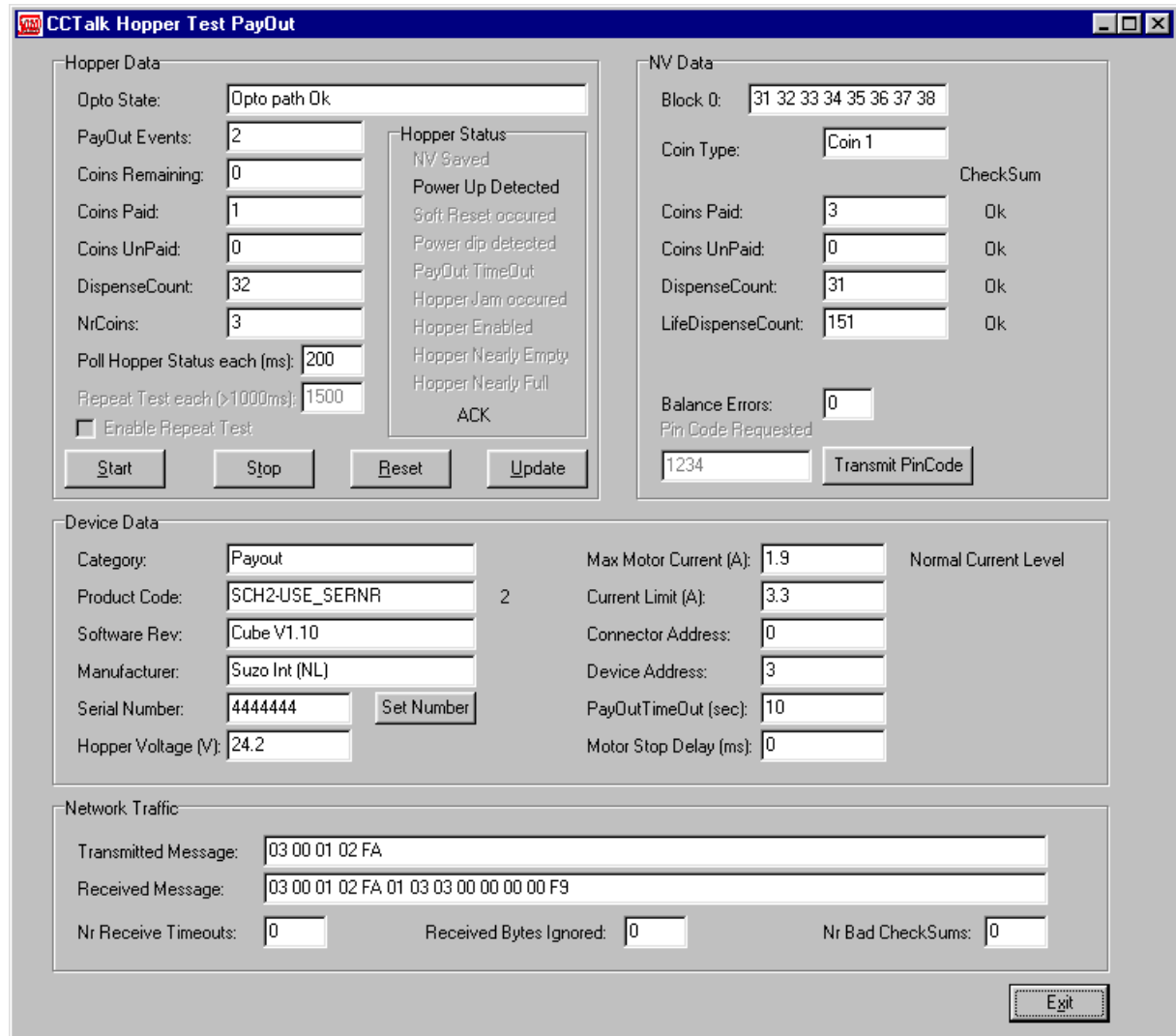
The sentence "Cipher Key received?" means if the test program has requested the serial number from the hopper using step 3.

### Example2: Full Hopper Test

Instead of controlling the hopper command by command using the command buttons on the screen, the hopper can be controlled in a more easy way using the Full Test button.



Pressing the Full Test button transmits almost all hopper commands at once. All received data is displayed on the screen as shown below:



The screen is divided into 4 sections:

- Hopper Data
- NV Data
- Device Data
- Network Traffic

### Section Hopper Data

This section shows all payout related data.

When the Start button is pressed, the hopper starts paying out the number of coins in the "NrCoins" field. This number can be changed by the user.

Each time a payout is started, all coin counters are updated automatically. This is done by transmitting the Hopper Status command at regular intervals. This interval time can be set in in the field "Poll Hopper Status each (ms)". Default is 200ms. The minimum poll time is 100ms.

Pressing the Stop button immediately stops the hopper, no matter if there is a coin in the coin exit port or not.

If there is a coin in the coin exit port, the hopper will not accept a new Start command, because it has its Opto State set to: "Opto blocked during idle". The Opto sensor is checked 3 times per second. If it sees a coin in the coin exit port, it will block any further attempts to payout.

Pressing the Reset button saves all running coin counter data in Non-Volatile memory. It also resets any opto error flags. If the "Opto blocked during idle" flag is set, a new payout can be started by pressing the Start button within 333ms after the Reset command. A new payout is started (since no opto blocking is detected yet) pushing out any pending coins in the coin exit port.

Pressing the Update button refreshes all Hopper data and NV (Non-Volatile) memory data.

### **NV Data section**

This section shows all EEPROM data.

Refer to Table 12: EEPROM Memory Description for more details.

The "Balance Errors" field contains the number of balance errors. After each Hopper Status command, the number of CoinsRemaining + NrCoinsPaid must be equal to NrCoins. If the balance is not correct this counter is incremented.

If the Pin code security mechanism is enabled, the user may enter the pin code in this field. Pressing Transmit PinCode transmits the pin code to the hopper.

### **Device Data section**

This section shows all Hopper Device data.

Refer to the relevant hopper commands for more details.

### **Network Traffic section**

This section shows all data traffic on the cctalk bus. It can be used to analyze all hopper commands and responses.

### **Acknowledgement**

This Test Program is delivered "as is". It is tested on Windows98 and Windows2000 Professional platform on a Pentium II 400MHz machine with 128MB Ram. It should be used as a tool to get the cctalk hopper up and running and it can be helpful for software developers who are writing the hopper drivers on their machines. It may not be used for any commercial purposes without written permission of Suzo International (NL) BV.

Design and specifications are subject to change without notice.  
Wijzigingen in ontwerp en technische gegevens voorbehouden, zonder kennisgeving.  
La conception et les spécifications sont modifiables sans préavis.  
El diseño y especificaciones están sujetos a cambios sin previo aviso.

This manual is intended only to assist the reader in the use of this product and therefore Suzo International shall not be held liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any omission from this manual or any incorrect use of the product.

**WARNING!**

Failure to observe the interface requirements specified in this technical manual may result in miscounts, damage to the electronics and the motor of the hopper or create unacceptable voltage drops, affecting other units depending on the same powersupply.



SUZO INTERNATIONAL (NL) B.V.  
Antonie van Leeuwenhoekstraat 9  
3261 LT Oud-Beijerland The Netherlands  
TEL: +31 (0)186 64.33.33 \* FAX: +31 (0)186 64.33.22