V² Pelicano

ccTalk interface

Author: **C.-P. Heins**

Table of contents

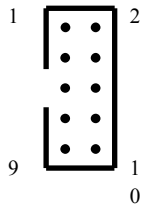
1 History

Date	*Version*	*Changes*
11.10.2010	1.0	First official version
16.11.2010	1.1	Errorcode table updated
26.08.2011	1.2	Accept Limit function, Unbuffered Mode
18.01.2012	1.3	Extended accept limit function, Operation with sorter
11.01.2013	1.4	Suggested error recovering added
23.05.2013	1.5	Further test functions added (header 255), optional validation event

2 General

2.1 ccTalk Connector, 10 pin

PIN	*Parallel*
1	Data
2	Data Gnd
3	n.c.
4	n.c.
5	n.c.
6	n.c.
7	Vdd (12-24 DC)
8	GND
9	n.c.
10	n.c.



2.2 Serial Protocol

2.2.1 Voltage levels

Mark state (idle) +3.3V nominal Range 2.0V to 5.0V*
Space state (active) 0V nominal Range 0.0V to 1.0V

* Input is 5V tolerant, Output is 3.3V

2.2.2 Format

9600 baud, 1 start bit, 8 data bits, no parity bit, 1 stop bit
The rise and fall time at 9600 baud should be less than 10us

2.2.3 Bus address

The standard bus address is 2.
It can be factory programmed to any value from 2 to 250.

2.2.4 Message Structure

The Pelicano uses the Standard structure with simple checksum. The Format with CRC is not supported.

For a payload of N data bytes…

[Destination Address]
[No. of Data Bytes]

[Source Address]
[Header]
[Data 1]
...
[Data N]
[Checksum]

Each communication sequence (a command or request for information) consists of 2 message packets structured in the above manner. The first will go from the master device to the slave device and then a reply will be sent from the slave device to the master device. The reply packet could be anything from a simple acknowledge message to a stream of data.

Note that the acknowledge message in cctalk conforms to the above structure in the same way all other messages do. Some protocols use a single byte acknowledge - this is not viewed as secure.

The structure does not alter according to the direction of the message packet. The serial protocol structure does not care who originates the message and who responds to it.

For a simple command or request with no data bytes…

[Destination Address]
[0]
[Source Address]
[Header]
[Checksum]

The acknowledge message is produced by setting the header to zero and having no data bytes…

[Destination Address]
[0]
[Source Address]
[0]
[Checksum]

For more detailed information refer to the official ccTalk specification downloadable at www.cctalk.org

3 Commandset overview

Table 3.1: Implemented commands

Header	***Command***	***Group***
255	Factory setup and test	
254	Simple poll	1
249	Request polling priority	C
248	Request status	C
247	Request variable set	C
246	Request manufacturer id	1
245	Request equipment category id	1
244	Request product code	1
243	Request database version	C
242	Request serial number	2
241	Request software revision	2
240	Test solenoids	C
239	Operate motors	B
238	*Test output lines*	*C*
237	*Read input lines*	*C*
236	Read opto states	C
232	Perform self-check	C
231	Modify inhibit status	C
230	Request inhibit status	C
229	Read buffered credit or error codes	C
228	Modify master inhibit status	C
227	Request master inhibit status	C
192	Request build code	1
184	Request coin id	C
4	Request comms revision	2
1	Reset device	2

4 Commandset details

4.1 Header 255 - Factory set-up and test

Transmitted data : <variable>
Received data : ACK or <variable>

This command is reserved for functions needed during the manufacture of a product and is not intended for general use. Every manufacturer can define their own set of functions buried behind this header number.

Commands for testing components
['B']['V'][1][Test ID]

Test ID:

0x10 = Diskmotor off
0x11 = Diskmotor forward
0x12 = Diskmotor reverse
0x13 = change disk calibration value (temporarily)
0x15 = move disk to stop position

0x20 = Trashmotor off
0x21 = Trashmotor forward
0x22 = trashmotor reverse
0x23 = change trashmotor calibration value (temporarily)
0x24 = trashdoor small cycle (trashdoor is opened and closed)
0x25 = trash complete cycle (trashdoor is opened, the disk turns, trashdoor is closed)

with firmware version 00.40 and above:
0x01 = Read status (bit0 = diskmotor active, bit1 = trashmotor active)
0x26 = pass-through position (trashdoor is opened and disk is moved to pass-through position)
0x27 = ready position (trashdoor is closed and disk is moved to stop position)

4.2 Header 254 - Simple poll

Transmitted data : <none>
Received data : ACK

This command can be used to check that the slave device is powered-up and working. No data is returned other than the standard ACK message and no action is performed. It can be used at

EMS (Early Morning Start-up) to check that the slave device is communicating. A timeout on this command indicates a faulty or missing device, or an incorrect bus address or baud rate.

4.3 Header 249 - Request polling priority

Transmitted data : <none>
Received data : [units] [value]

This is an indication by a device of the recommended polling interval for buffered credit information. Polling a device at an interval longer than this may result in lost credits.

[units]
0 = special case
1 = ms
2 = x 10 ms

The Pelicano will answer with [2][20] which means 200 ms.

4.4 Header 248 - Request status

Transmitted data : <none>
Received data : [status]

This command reports the status of the BV. Because the BV has non of the documented error states the answer will always be [0]

4.5 Header 246 - Request manufacturer id

Transmitted data : <none>
Received data : "NRI"

4.6 Header 245 - Request equipment category id

Transmitted data : <none>
Received data : "Coin Acceptor"

4.7 Header 244 - Request product code

Transmitted data : <none>
Received data : "BV"

The product code is returned. No restriction on format.

The complete product identification string can be determined by using ‘Request product code’ followed by ‘Request build code’.

4.8 Header 243 - Request database version

Transmitted data : <none>
Received data : [calibration database no.]

This command retrieves a database number from 1 to 255 which may be used for remote coin programming.

A database number of 0 indicates remote coin programming is not possible.

4.9 Header 242 - Request serial number

Transmitted data : <none>
Received data : [serial 1] [serial 2] [serial 3] [serial 4]

serial 1 = LSB
decimal = [serial 1] + 256 * [serial 2] + 65536 * [serial 3] + ...

4.10 Header 241 - Request software revision

Transmitted data : <none>
Received data : "mm.ss“

mm = main revision
ss = sub revision

4.11 Header 240 - Test solenoids

Transmitted data : [bit mask]
Received data : ACK

[bit mask] = not used
All solenoids are pulsed for 200 ms one after another. The bit mask makes no difference. If a sorter is mounted its solenoids will be pulsed also.

The slaves ACK is returned before the procedure is started.

4.12 Header 239 - Operate motors

Transmitted data : [Command] <var data>
Received data : ACK or <var data>

[Command]
1 = trash cycle 1 (trashdoor is opened, the disk turns, trashdoor is closed)
---> works only when device is disabled (Master Inhibit)
2 = trash cycle 2 (for future use)
3 = trash cycle 3 (for future use)
4 = issue CPR start (starts disk in Start-on-CPR-Mode although CPR is inactive)
---> works in Start-on-CPR-Mode only. Device must be enabled (Master-Inhibit)
10 = set speed (one data byte needed)
data = speed value in percent. 100 = 3 coins/s, 133 = 4 coins/s
NOTE: This changes the speed temporarily. After a reset this value is always 100%.
11 = get speed (one data byte returned)
data = speed value in percent.
Returns the actual speed value.
12 = get pocket time (returns one byte of data)
data = pocket time [in 4 ms steps]
If the disk turns the time from one pocket to the next is measured. This value can be used to determine the disk speed in coins/s. NOTE: only use this command if no coins are in the container, otherwise the returned value is not valid.

4.13 Header 236 - Read opto states

Transmitted data : <none>
Received data : [bit mask]

Checks the optical sensors at the coin exit.

[bit mask]
Bit 0: Coin Present Sensor (1 = active, there is at least one coin within the container)
Bit 1: Trashdoor (1 = open, 0 = closed)
Bit 2: Lower Sensor (Accept & Reject Path)

Bit 3: Upper Sensor (Accept path) (0 = opto clear, 1 = opto blocked)

4.14 Header 232 - Perform self-check

Format (a)
Transmitted data : <none>
Received data : [fault code]

0 = OK
1 = firmware checksum corrupted
2 = Fault on inductive coils (measurement system)
3 = fault on credit sensors
30 = Datablock checksum corrupted
253 = coin jam in measurement system
254 = Disk Blocked (Disk is blocked, device was not able to resolve blockage)
255 = unspecified alarm code

4.15 Header 231 - Modify inhibit status

Transmitted data : [inhibit mask 1] [inhibit mask 2]
Received data : ACK

This command sends an individual inhibit pattern to the Pelicano.
With a 2 byte inhibit mask, up to 16 coins can be inhibited or enabled.

[inhibit mask 1]
Bit 0 - coin 1
...
Bit 7 - coin 8

[inhibit mask 2]
Bit 0 - coin 9
...
Bit 7 - coin 16

0 = coin disabled (inhibited)
1 = coin enabled (not inhibited)

NOTE: This setting is temporary (RAM). After a device reset all coins are inhibited

4.16 Header 230 - Request inhibit status

Transmitted data : <none>
Received data : [inhibit mask 1] [inhibit mask 2]

This command requests an individual inhibit pattern from the Pelicano.

See 'Modify inhibit status' for more details.

4.17 Header 229 - Read buffered credit or error codes

Transmitted data : <none>
Received data : [event counter]
[result 1A] [result 1B]
[result 2A] [result 2B]
[result 3A] [result 3B]
[result 4A] [result 4B]
[result 5A] [result 5B]

This command returns a past history of event codes for a coin acceptor in a small data buffer. This allows a host device to poll a coin acceptor at a rate lower than that of coin insertion and still not miss any credits or other events.

The standard event buffer size is 10 bytes which at 2 bytes per event is enough to store the last 5 events.

A new event ripples data through the return data buffer and the oldest event is lost.

For example, consider a 5 event buffer :

result 5A ==> lost
result 5B ==> lost

result 4A ==> result 5A
result 4B ==> result 5B

result 3A ==> result 4A
result 3B ==> result 4B

result 2A ==> result 3A
result 2B ==> result 3B

result 1A ==> result 2A
result 1B ==> result 2B

new result A ==> result 1A
new result B ==> result 1B

An event counter is used to indicate any new events and this must be compared at each poll to the last known value.

event counter - last event counter	Stored in result…
0	-
1	1
2	1, 2
3	1, 2, 3
4	1, 2 ,3, 4
5	1, 2, 3, 4, 5
6+	1, 2, 3, 4, 5 & others are lost

[event counter]
0 (power-up or reset condition)
1 to 255 - event counter

[result A]
1 to 255 - credit
0 - error code

[result B] for credits
1 – coin was accepted to path 1 (default / without sorter)
2 – coin was accepted to path 2
3 – coin was accepted to path 3
…
8 – coin was accepted to path 8

100 - Validation Event (configurable option), valid & enabled coin was measured

[result B] for error codes
refer to Section 5 „Errorcodes“

The event counter is incremented every time a new credit or error is added to the buffer. When the event counter is at 255 the next event causes the counter to change to 1. The only way for the counter to be 0 is at power-up or reset. This provides a

convenient signalling mechanism to the host machine that a major fault has occurred.

Validation event for machine controlled sorter:
If the machine has to control a sorting device the standard credit event comes too late to switch the sorter.
In this case an optional validation event can be used to get earlier information on the coin. The validation event is generated after a valid and enabled coin is measured. This is almost 200ms ahead of the credit event. With this option activated there will be two events for each coin.
NOTE: Due to vibrations a coin may not leave the disk after it was measured. It stays in the container for another round and will get measured again. In this case there will be only the validation event and the corresponding credit event will be missing. The coins must not be credited on validation events.

4.17.1 Static errors
For static errors like 'credit sensor blocked' there will be a NULL-event after the error is solved.

4.17.2 Coin acception
The Pelicano will credit a coin when it has left through the acceptance exit. The coin must not be obstructed while leaving the device.

4.17.3 Automatic inhibition
The device will stop accepting coins for the following reasons:

1. The device was not polled (Header 229) for more than 500ms
2. The event buffer is full (contains more than 5 events)

If the Pelicano is enabled it will automatically disable itself because it considers the machine controller to be down.

4.17.4 Unbuffered Mode
This mode gives the host full control on acception rates. If the host needs to delay the acception of the next coin for some reason (e.g. to rotate a storage container) this mode should be used.

- Enabled by device settings (production)
- After every accepted coin the acceptance is temporarily inhibited until the ccTalk event buffer is read twice (header 229)
- The first reading transfers the coin info. The second reading re-enables the acception.
- Between the first and the second reading the host can change sorter settings and/or inhibit mask.

- During temporarily inhibition the disk will continue turning while coins are kept in the container.

4.18 Header 228 - Modify master inhibit status

Transmitted data : [XXXXXXX | master inhibit status]
Received data : ACK

[master inhibit status]
Bit 0 only is used.
0 – Pelicano is disabled (disk stops movement)
1 – Pelicano is enabled (disk starts turning, coins will be processed)

NOTE:
After a reset the Pelicano is disabled.
In an optional configuration the BV will not stop immediately if it is disabled. It will stop acception immediately but continues turning the disk until the container is empty. At least this will take 3 seconds.

4.19 Header 227 - Request master inhibit status

Transmitted data : <none>
Received data : [XXXXXXX | master inhibit status]

[master inhibit status]
Bit 0 only is used.
0 – Pelicano is disabled (disk stops movement)
1 – Pelicano is enabled (disk starts turning, coins will be processed)

4.20 Header 222 - Modify sorter override status

Transmitted data : [sorter override bit mask]
Received data : ACK

[sorter override bit mask]
B0 - Sorter Path 1
...
B7 - Sorter Path 8

0 = sorter override to a different or default path
1 = no action, normal sorting

This command allows the sorter override status to be set in a coin acceptor. Each bit represents a sorter path for the accepted coin. A zero overrides that sorter path to another one (possibly the default sorter path).

Settings are temporary (stored in RAM). After a reset no overrides are active (all 1s).

4.21 Header 221 - Request sorter override status

Transmitted data : <none>
Received data : [sorter override bit mask]

This command returns the sorter override status in a coin acceptor. Each bit represents a sorter path for the accepted coin. A zero means that the sorter path has an active override.

Refer to the 'Modify sorter override status' command for more details.

4.22 Header 213 - Request option flags

Transmitted data : <none>
Received data : [bit7...bit0]

[bit4]
Bit 4 only is used.
0 – accept limit feature not supported
1 – accept limit feature supported

4.23 Header 210 – Modify sorter paths

Format (a)
Transmitted data : [coin position] [path]
Received data : ACK

Format (b)
Transmitted data : [coin position] [path] [ov. path 1] [ov. path 2] [ov. path 3]
Received data : ACK

This command modifies the sorter path(s) for each coin position.

Format a changes the primary path only and format b changes the override paths also. The settings are temporary (stored in RAM). After a reset a factory setting is loaded. The factory setting can be changed via configuration equipment. (e.g. Heartbeat)

[coin position]
1 to 16

[sorter path]
1 to 8

4.24 Header 209 – Request sorter paths

Format (a) (ACMII version)
Transmitted data : [coin position]
Received data : [path]

Format (b) (Standard version)
Transmitted data : [coin position]
Received data : [path] [ov. path 1] [ov. path 2] [ov. path 3]

This command allows sorter paths to be requested in a coin acceptor.

See the 'Modify sorter paths' command for more details.

4.25 Header 192 - Request build code

Transmitted data : <none>
Received data : ASCII

The product build code is returned. No restriction on format.

'IT0' = Italian version (ACMII)
'IT1' = Italian version (ACMII), write protected
'DE0' = Standard version
'DE2' = Standard version, write protected

If a device is write protected it cannot be reconfigured in the field.

4.26 Header 189 - Modify default sorter path

Transmitted data : [default path]
Received data : ACK

[default path]
1 to 8

This command allows the default sorter path on a coin acceptor to be changed. If there is an active override on the current coin sorter path then it will be routed to the default path.

Changes are temporary (stored in RAM).

4.27 Header 188 - Request default sorter path

Transmitted data : <none>
Received data : [default path]

This command reads the default sorter path on a coin acceptor.

See the 'Modify default sorter path' command for more details.

4.28 Header 184 - Request coin id

Transmitted data : [coin position]
Received data : [char 1] [char 2] [char 3]…

Each coin position, for example 1 to 16, is interrogated for an ASCII identifier. This consists of 6 characters representing the coin name.

The identifier is made up as follows…

[C] [C] [V] [V] [V] [I]

CC = Standard 2 letter country code e.g. GB for the U.K. (Great Britain)
VVV = Coin value in terms of the base unit appropriate to that country
I = Mint Issue. Starts at A and progresses B, C, D, E…

The country code for the 'Euro', the Common European currency, is 'EU'.

If the country code is 'TK' then a token occupies this position rather than a coin. In this case the VVV field represents a token number in ASCII rather than a value which could change from one jurisdiction to another.

It is possible to have more than one mint issue in circulation at any particular time - for instance during a transition period from 'old' coins to 'new' coins. Serial coin acceptors can be programmed with both types and the 'old' coins inhibited by the host

machine when they officially go out of circulation.

Examples:
EU100A = 1,00 Euro
CA005B = 0,05 Canadian Dollar

4.29 Header 135 – Set/Get Accept Limit

a) Set Limit

Transmitted data : [mode] [n1] [n2] [n3]
Received data : none

limit = [n1] + 256 * [n2] + 65536 * [n3]
The accept limit is given in pricipal subdivisions. E.g, cents for Euro, 1359 = 13,59€

b) Get Limit

Transmitted data : none
Received data : [mode] [n1] [n2] [n3]

remaining limit = [n1] + 256 * [n2] + 65536 * [n3]

The remaining limit will be transmitted. This is the limit that was set before minus the coins accepted so far.

If an accept limit was set, the Pelicano will accept coins until the limit is reached. Further coins will be rejected.
The mode parameter chooses the behavior for the accept limit function:

4.29.1 Mode 0 = switch off
0 = no accept limit (normal function). This is to switch off an accept limit that was set before.

4.29.2 Mode 1 = overpaying allowed
The Pelicano tries to avoid overpaying by circulating coins within the container to check if there are coins with appropriate values. After the first cycle of the disk without detecting an appropriate coin, the pelicano does another cycle to accept that coin with the lowest value of the residual ones. The rest is rejected.

4.29.3 Mode 2 = no overpaying
Overpaying not allowed. Device will reject all none suitable coins.

4.30 Header 4 - Request comms revision

Transmitted data : <none>
Received data : [1] [4] [4]

4.31 Header 1 - Reset device

Transmitted data : <none>
Received data : ACK

This command forces a soft reset in the slave device. It is up to the slave device what action is taken on receipt of this command and whether any internal house-keeping is done. The action may range from a jump to the reset vector to a forced physical reset of the processor and peripheral devices. This command is worth trying before a hard reset (or power-down where there is no reset pin) is performed.

The slave device should return an ACK immediately prior to resetting and allow enough time for the message to be sent back in full.

The host device should wait at least 300ms after issuing a ‘Reset device’ command before sending the next command.

5 Errorcodes

cctalk No	*cctalk Error*	*Description*	*Coin Rejected*	*Suggested recovering method*
0	No Error, Null Event	No error, static error resolved	no	-
1	RejectCoin	Coin was rejected because it was unknown or master inhibit is active	yes	-
2	InhibitedCoin	Valid (but inhibited) coin rejected	yes	-
5	ValidationTimeout	If the disk gets blocked while a coin is within the measurement area (not critical)	possible	-
6	CreditSensorTimeout	A valid coin didn't reach the credit sensor within 500ms. This can have two possible reasons: 1) (critical) coin got stuck on its way to the credit sensor 2) (not critical) coin didn't fall off the disk and is still in the container. In this case it will be validated again and the error can be ignored.	possible	count
8	2ndCloseCoinError	There are 2 coins in one pocket, both will be rejected	yes	observe
10	CreditSensorNotReady	Coin was rejected because credit sensor is busy detecting previous coin(s) or is blocked	yes	observe
14	CreditSensorBlocked (Static Error)	Credit sensor is permanently blocked	no	Wait & reset
15	Sorter opto blocked (Static Error)	One or more optical sensors blocked on external sorter.	no	Wait & reset
29	AcceptGateOpenNotClosed	Unknown or inhibited coin was accepted (acceptance gate stuck)	no	stop
30	AcceptGateClosedNotOpen	Valid coin was rejected (acceptance gate stuck)	yes	observe
*117**	*Double signal on coin exit*	*Jumping coin. One coin is generating multiple signals on coin exit sensor. This is only informational but it points to a problem with coin guidance after the pelicano exit. Coins may jam after they left the pelicano.*	*no*	*Observe/ ignore*
*118**	*Disk stalled (Static Error)*	*Blocking could not be solved. Diskmotor was switched off. Device was disabled. Use Master Inhibit command to reenable device.*	*no*	*stop*
*119**	*Diskmotor overcurrent*	*Overcurrent was detected. Disk is reversing to solve the blocking.*	*no*	-
*120**	*External Light (Static Error)*	*Interfering light on CP or measurement sensors detected*	*no*	*Wait & reset*
*121**	*Validation Sensor Blocked (Static Error)*	*Coin Jam in validation area or validation sensor needs cleaning*	*no*	*Wait & reset*
254	CoinReturnMechanism		no	Wait &

cctalk No	*cctalk Error*	*Description*	*Coin Rejected*	*Suggested recovering method*
	Activated (Static Error)			reset
255	UnspecifiedAlarmCode		no	

* = NRI-specific (not part of offical cctalk specification)

5.1 Suggested recovering methods

The following is only a proposal because the recovering method is very application dependent. It may be sensible to ignore some errors while others will be regarded more critical.

count
Count this error. Every successfully accepted coin clears this counter.
Counter > 2 → set machine out of order
otherwise = ignore

wait&reset
Stop coin input.
If the error stays active for more than 10 seconds:

- reset the pelicano and see if it recovers
- if the error is still there put the machine out of order.

stop
Fatal error ---> Set the machine out of order / Call the attendant

observe
possible mechanical problem developing. Ignore if reported once.
If error count gets higher than 2% compared with no. of accepted coins → out of order

6 Operation

To operate the Pelicano the minimum steps are:

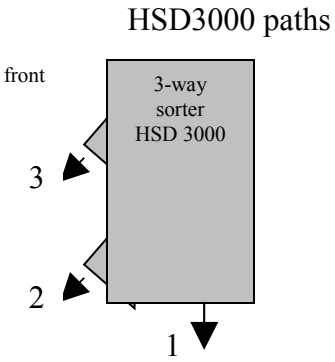
1. appropriate power connection 12..24V (+/- 10%), >= 500mA
2. Enable one or more coins (Header 231)
3. poll the device (Header 229) and process events/errors
4. repeatedly request coin present sensor (CPS) (Header 236)
5. if CPS is inactive proceed with step 3
6. Enable device (Header 228)
7. repeatedly poll the device and process events (Header 229) and where required request CPS state (Header 236)
8. if price is reached or if coin present sensor is inactive and there was no coin validated for at least 3 seconds disable device and proceed with step 3
9. proceed with step 7

For the loops within the 9 steps (polling) we recommend a period of 200 ms.

All other commands can be used to request more information (e.g Programmed coins). They are not necessary to operate the Pelicano.

7 Operation with sorter

The pelicano is capable of controlling a HSD3000 sorting device. Accepted coins can be routed to three different paths (1-3).



After a reset the sorter setting is loaded with factory programmed values.
If no changes are necessary during operation there is no need for the host machine to use any sorter command. Otherwise it can gain full control by using the ccTalk headers 188, 189, 209, 210, 221 and 222.
The machine is allowed to change a path at any time but a coin being processed when a change takes place will probably be routed to the old (unchanged) path.

7.1 Operation with primary path only

The simplest way of operating the sorter is to modify the primary path. In this case the primary path for each coin is set as needed. Implementing the headers 209 and 210 is sufficient.

During startup:
Set primary path for each coin. (header 210)

During operation:
Modify primary paths (header 210)

7.2 Operation with override paths

In addition to the primary path, up to three override paths can be defined for each coin. During operation the machine will only mark one or more path(s) as 'full' (override). If a coin's primary path is marked as full, the coin will be routed to the next 'free' override path. If all override paths are full the coin will be routed to the default path.

During startup:

Set primary and override paths for each coin. (header 210)
Set default sorter path (header 189)

During operation:

Modify override status (header 222)

Example:

coin 3: 0,20€ paths=2,3,5,5
default path = 1

paths marked as full:	*coin 3 routed to*
no	*path 2 (primary path)*
path 2	*path 3*
path 3	*path 2 (primary path is still 'free')*
path 2+3	*path 5*
path 2+3+5	*path 1 (default path)*