V² Eagle

ccTalk interface

Author: **Claus-Peter Heins**

Table of contents

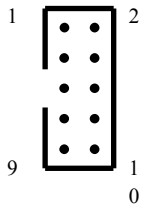
1 History

Date	*Version*	*Changes*
18.12.2013	0.1	Preliminary version
06.08.2014	0.2	SW 504 added, configuration option '32 coin positions' added
18.11.2015	0.3	Section Error codes revised, Section sorter override mechanism added, Section Delayed activation of settings added.
29.11.2016	0.4	More detailed description for Header 140 (Upload firmware)

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 Eagle 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 Command set overview

The following table provides on overview which commands are available for different firmware versions. It is also stated which commands are configuration dependent and which are implemented for compatibility reasons without functionality.

Header	Command	FW430	FW445 FW504	FW495
255	Factory set-up and test	✓	✓	✓
254	Simple poll	✓	✓	✓
253	Address poll	✓	✓	✓
252	Address clash	✓	✓	✓
251	Address change	A	A	A
250	Address random	A	A	A
249	Request polling priority	✓	✓	✓
248	Request status	✓	✓	✓
247	Request variable set	✓	✓	✓
246	Request manufacturer id	✓	✓	✓
245	Equipment category id	✓	✓	✓
244	Request product code	✓	✓	✓
243	Request database version	✓	✓	✓
242	Request serial number	✓	✓	✓
241	Request software revision	✓	✓	✓
240	Test solenoids	✓	✓	✓
236	Read opto states	✓	✓	✓
232	Perform self-check	✓	✓	✓
231	Modify inhibit status	✓	✓	✓
230	Request inhibit status	✓	✓	✓
229	Read buffered credit or error codes	✓	✓	✓
228	Modify master inhibit status	✓	✓	✓
227	Request master inhibit status	✓	✓	✓
226	Request insertion counter	-	(✓)	✓
225	Request accept counter	-	(✓)	✓
222	Modify sorter override status	✓	✓	✓

Header	Command	FW430	FW445 FW504	FW495
221	Request sorter override status	✓	✓	✓
219	Enter new PIN number	✓	✓	✓
218	Enter PIN number	✓	✓	✓
216	Request data storage availability	(✓)	✓	✓
215	Read data block	-	✓	✓
214	Write data block	-	✓	✓
213	Request option flags	✓	✓	✓
210	Modify sorter paths	✓	✓	✓
209	Request sorter paths	✓	✓	✓
202	Teach mode control	T	T	T
201	Request teach status	T	T	T
197	Calculate ROM checksum	-	✓	✓
196	Request creation date	-	✓	✓
195	Request last modification date	-	✓	✓
194	Request reject counter	-	(✓)	✓
193	Request fraud counter	-	(✓)	(✓)
192	Request build code (with write protection Info)	✓	✓	✓
189	Modify default sorter path	✓	✓	✓
188	Request default sorter path	✓	✓	✓
184	Request coin id	✓	✓	✓
181	Modify security setting	✓	(✓)	(✓)
180	Request security setting	✓	(✓)	(✓)
179	Modify bank select	-	(✓)	(✓)
176	Request alarm counter	-	(✓)	(✓)
173	Request thermistor reading	-	(✓)	(✓)
170	Request base year	-	✓	✓
169	Request address mode	-	✓	✓
165	Modify variable set	-	(✓)	(✓)
162	Modify inhibit and override registers	-	✓	✓
150	Request individual accept counter	-	(✓)	(✓)
141	Request firmware upgrade capability	-	✓	✓

Header	Command	FW430	FW445 FW504	FW495
140	Upload firmware	-	✓	✓
139	Begin firmware upgrade	-	✓	✓
138	Finish firmware upgrade	-	✓	✓
111	Request encryption support	-	✓	✓
96	Remote coin programming	-	R	R
83	Disable intelligence	-	(✓)	(✓)
74	Calculate configuration checksum	-	✓	-
4	Request comms revision	✓	✓	✓
3	Clear comms status variables	-	(✓)	(✓)
2	Request comms status variables	-	(✓)	(✓)
1	Reset device	✓	✓	✓

✓ = fully supported
(✓) = command without functionality
A = supported with option “address change” enabled
R = supported with option “remote coin programming” enabled
T = supported with option “remote teach” enabled

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.

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 253 – Address poll

Refer to the section MDCES in the official ccTalk specification.

4.4 Header 252 – Address clash

Refer to the section MDCES in the official ccTalk specification.

4.5 Header 251 – Address change

Refer to the section MDCES in the official ccTalk specification.
This command is only working if the Eagle is equipped with the option “Address change enable”.

4.6 Header 250 – Address random

Refer to the section MDCES in the official ccTalk specification.
This command is only working if the Eagle is equipped with the option “Address change enable”.

4.7 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 Eagle will answer with [2][20] or with [1][200] which means 200 ms.

4.8 Header 248 - Request status

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

0 = OK
1 = Coin return mechanism activated (flight deck open)
2 = Coin on string mechanism was activated. Device is temporarily disabled.

4.9 Header 247 - Request variable set

Format (a)
Transmitted data : <none>
Received data : [n1][n2][n3][n4]

The four bytes returned represent the customer this device was manufactured for.
Customer id = [n1] + 256 * [n2] + 65536 * [n3] + ...

NOTE: There may be further formats being defined in the future

4.10 Header 246 - Request manufacturer id

Transmitted data : <none>
Received data : "NRI"
“Money Controls” (→ SR5 compatibility mode = 01h)

4.11 Header 245 - Request equipment category id

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

4.12 Header 244 - Request product code

Transmitted data : <none>
Received data : "EAGLE"
“SR5i” (→ SR5 compatibility mode = 02h)

4.13 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.

4.14 Header 242 - Request serial number

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

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

4.15 Header 241 - Request software revision

Transmitted data : <none>
Received data : "xxxyyzz“ (7 digits)

xxx = base version (000...999)
yy = release (00..99)

zz = revision (00..99)

4.16 Header 240 - Test solenoids

Transmitted data : [bit mask]
Received data : ACK

[bit mask]:
00h = tests all gates one after another
01h = Gate 0 (accepting)
02h = Gate 1 (internal sorter)
04h = Gate 2 (internal sorter)
08h = Gate 3 (internal sorter/external manifold sorter)
10h = upper gate (external sorter)
20h = lower gate (external sorter)

The bits may be ORed to test more than one gate at once. But caution: this will increase the supply current.

The slaves ACK is returned before the procedure is started.

4.17 Header 236 - Read opto states

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

Checks the internal and external optical sensors.

[bit mask]
Each bit within the bit mask reflects one or more physical sensors.
0 = OK, 1 = blocked/faulty

Bit	*FW 430*	*FW 445/496/504*
0 (01h)	TF1 / SO1* / SP1**	EN
1 (02h)	TF2 / SO2* / SP2**	0
2 (04h)	TF3 / SO3* / SP3**	EX | CP4 | CP5 (combined sorter optos)

Bit	*FW 430*	*FW 445/496/504*
3 (08h)	TF4	SO1 | SO2 | SO3
4 (10h)	CP3	CP3
5 (20h)	CP4	CP4
6 (40h)	0	CP5
7 (80h)	0	SP1 | SP2 | SP3

SO = sizing optics
SP = sorter position sensors (assembly option)
TF = tube full optos (right/mid-right/mid-left/left), only on 5-way cross sorter
CP = internal coin position sensors
EN = optional external coin entry opto
EX = optional external manifold opto

* if device reports fault code 6 (fault on sizing optics), bit1..3 will reflect the sizing optics (SO1..3)
** if device reports error 15 (Diverter opto blocked), bit 1..3 will reflect the sorter position sensors

4.18 Header 232 - Perform self-check

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

Format (b)
Transmitted data : <none>
Received data : [fault code] [extra info]

This command instructs the peripheral device to perform a self-check. Where more than one fault exists on a product, faults will be reported in priority order. Once one fault is fixed, the next fault will be reported.

Refer to section 6 Fault codes on page 44 for a description of possible fault codes and their meaning.

NOTE: This command decodes the internal error flag register and will answer immediately. It should be called repeatedly but less often than header 229. We recommend every second.

4.19 Header 231 - Modify inhibit status

Format (a) – 16 coin positions
Transmitted data : [inhibit mask 1] [inhibit mask 2]
Received data : ACK

Format (b) – 32 coin positions
Transmitted data : [inhibit mask 1] [inhibit mask 2] [inhibit mask 3] [inhibit mask 4]
Received data : ACK

This command sends an individual inhibit pattern to the Eagle.
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

[inhibit mask 3]
Bit 0 - coin 17
...
Bit 7 - coin 24

[inhibit mask 4]
Bit 0 - coin 25
...
Bit 7 - coin 32

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

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

NOTE: If the Eagle is configured to “32 coin positions” mode and a format (a) inhibit mask is received, it will duplicate the data internally.
[inhibit mask 3] = [inhibit mask 1] and [inhibit mask 4] = [inhibit mask 2]
This allows to enable/disable all 32 coins with a short (standard) command.

4.20 Header 230 - Request inhibit status

Format (a) – 16 coin positions
Transmitted data : <none>
Received data : [inhibit mask 1] [inhibit mask 2]

Format (b) – 32 coin positions
Transmitted data : <none>
Received data : [inhibit mask 1] [inhibit mask 2] [inhibit mask 3] [inhibit mask 4]

This command requests an individual inhibit pattern from the Eagle.

See 'Modify inhibit status' for more details.

4.21 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

[result B] for error codes
refer to Section 5 Event codes

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

4.21.1 Static errors

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

4.21.2 Coin acception

The Eagle will credit a coin when it has left through the internal sorter exit. The coin must not be obstructed while leaving the device.

4.21.3 Automatic inhibition

The device will stop accepting coins for the following reasons:

1. The device was not polled (Header 229) for more than 500ms (5-way sorter)timeout is configurable from 500 to 2500 ms, Standard is 500)
2. The event buffer is full (contains more than 5 events)

This stop is only temporarily. If the host resumes polling the Eagle will resume accepting automatically.

4.21.4 Unbuffered Mode

This mode gives the host full control on accepting speed. If the host needs to delay the acception of the next coin for some reason (e.g. to rotate a storage container or change sorter settings) 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.

4.22 Header 228 - Modify master inhibit status

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

[master inhibit status]
Bit 0 only is used.

0 – Eagle is disabled (all coins inserted will be rejected)
1 – Eagle is enabled (coins inserted will be processed)

NOTE:
After a reset the Eagle is disabled.

4.23 Header 227 - Request master inhibit status

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

[master inhibit status]
Bit 0 only is used.
0 – Eagle is disabled (all coins inserted will be rejected)
1 – Eagle is enabled (coins inserted will be processed)

4.24 Header 226 - Request insertion counter

Transmitted data : <none>
Received data : [n1] [n2] [n3]

This command returns the total number of coins / bills put through a device.

24-bit counter in decimal = [n1] + 256 * [n2] + 65536 * [n3]
Range 0 to 16,777,215

The counters are stored in EEPROM and their value is retained after a power-down or reset. The counter value is stored every 5 insertions and so should be used as an approximate guide to performance rather than for auditing purposes.

4.25 Header 225 - Request accept counter

Transmitted data : <none>
Received data : [n1] [n2] [n3]

This command returns the total number of coins / bills accepted by a device.

24-bit counter in decimal = [n1] + 256 * [n2] + 65536 * [n3]
Range 0 to 16,777,215

The counters are stored in EEPROM and their value is retained after a power-down or reset. The counter value is stored every 5 insertions and so should be used as an approximate guide to performance rather than for auditing purposes.

4.26 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).

See also section 4.32.1 Sorter override mechanism for a more detailed discription.

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

4.27 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.

4.28 Header 216 - Request data storage availability

Transmitted data : <none>
Received data : [memory type]
[read blocks] [read bytes per block]
[write blocks] [write bytes per block]

Some slave devices allow host data to be stored for whatever reason. Whether this service is provided at all can be determined by using this command. For a more detailed description refer to the official ccTalk specification.

The v² Eagle will answer to this command depending on its configuration with one of the following.

[0] [0] [0] [0] [0] = data storage not supported
[2] [2] [8] [2] [8] = 2 blocks of 8 bytes each can be read and written from/to EEPROM

NOTE:
For the Italian flavor configuration this command is not answered.

4.29 Header 215 - Read data block

Transmitted data : [block number]
Received data : < variable >

[block no.]
0 to 255 (1st block number is always zero)

4.30 Header 214 - Write data block

Transmitted data : [block number] <variable>
Received data : ACK

[block no.]
0 to 255 (1st block number is always zero)

The return ACK from the slave device is only sent after a write cycle has been performed. It us up to the host software whether a verify operation is performed by reading back the data and comparing it.

4.31 Header 213 - Request option flags

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

[bit7]
0/1 – remote teach not supported/supported
If supported the host can use the headers 202 (Teach mode control) and 201 (Request teach status) to get a coin programmed into the validator.

[bit6..bit0]
not used (=0)

4.32 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

[coin position]
1 to 16 or 1 to 32

[sorter path]
1 to 8

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 will be stored in EEPROM.

To prevent the NVM from getting worn out, the sorter override mechanism (refer to section 4.32.1 Sorter override mechanism) should be used to override designated paths to cashbox instead of changing the path setting.

4.32.1 Sorter override mechanism

For every coin, four sorter paths can be defined. The validator will route the coin to the first path of the four that is not marked as overridden (full) by the sorter override mask.

Example:
coin 3: path setting = 3, 4, 5, 1
coin 4: path setting = 1, 1, 1, 1

1) override mask = 1111 1111b (no overrides)
coin 3 will be routed to path 3 and coin 4 to path 1

2) override mask = 1111 0011b (override path 3 and 4)
Coin 3 cannot be routed to 3 (it's full). The next alternative, path 4, is also full. Therefore the 3rd setting, path 5, will be used. Nothing will change for coin 4. It will still be routed to path 1.

4.33 Header 209 – Request sorter paths

Format (a) (Italian flavor 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.

4.34 Header 202 – Teach mode control

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

Format (b)
Transmitted data : [position] [orientation]
Received data : ACK

Format (c)
Transmitted data : [position] [orientation] [control] [code 1] [code 2] [code 3]
Received data : ACK

Format (d)
Transmitted data : [position] [orientation] [control] [insertions] [security]
Received data : ACK

[position]
This is the coin position to program using teach
e.g. 1 to 16 or 1 to 32

NOTE: All coin positions that are not factory programmed, can be used to teach coins. If the selected position is factory programmed, the next use of header 201 will report 'teach error'. A position already used for teaching can be overwritten at any time by another teach process. The position doesn't have to be erased beforehand.

[orientation]
0 – not used. For coins, the orientation parameter is ignored anyway.
1 to 4 - teaching bills requires the orientation to be known in advance. Refer to the product documentation for the orientation convention

[control]
1 – delete teach position
2 – teach with specified parameters

[code N]
For deleting teach positions, the data must be the 1's complement of the serial number, LSB first (3 or 4 bytes, as returned by the 'Request serial number' command). This guards against accidental deletion.

[insertions]
The number of coins required to complete the teach process. The more coins you insert, the more accurate the teach process and the better the accept rate will be.
Typical range 10 to 30.

[security]
0 – standard acceptance windows
1 – wide (better true coin accept rate but reduced fraud performance)

Configuration Requirements:
To make the remote teach available, the device must be ordered with "Remote Teach".
All coin positions that are not factory programmed, can be used to teach coins. If the selected position is factory programmed, header 201 will report 'teach error'.

Coin ID string:
Out of the box the Eagle will report '…...' for all available / unused coin positions. After the teaching process was successful the coin ID will be named 'XX000A' or 'TEACH ' (refer to option coin ID TEACH). If that position will be erased, the coin ID will change back to '…...'.

Command procedure:

For the Eagle the following procedure is mandatory:

1. put the device into teach mode (header 202)

2. read teach status (header 201)
3. repeat step 2 until:
   - teach status is 'teach completed' or
   - user wants to abort the teach process

4. NORMAL: do nothing
   ABORT: abort teach operation (header 201)
5. Because the teach mode changes the configuration, the Eagle will perform a software reset after a successful teach process. The host should at least read coin Ids again (header 184) and set the inhibit status (header 231)
6. switch back to normal operation

4.35 Header 201 – Request teach status

Format (a) – default
Transmitted data : [0]
Received data : [no. of coins / bills entered] [status code]

Format (b) – abort teach operation
Transmitted data : [1]
Received data : [no. of coins / bills entered] [status code]

[status code]
252 - teach aborted
253 - teach error
254 - teaching in progress (busy)
255 - teach completed

This command is used to monitor the progress of teach mode. Only when a 'teach completed' status message is received can the operation be deemed successful.

The actual teach mechanism is under the full control of the slave device. It decides when enough coins or bills have been entered.

NOTE:
If not otherwise specified, the Eagle expects 10 coins being entered. After the 10 coins are entered the data will be calculated and programmed into the internal flash memory. This process takes up to 1 second. During that time the device will not answer to any ccTalk command.

4.36 Header 197 – Calculate ROM checksum

Transmitted data : <none>
Received data : [chk1] [chk2] [chk3] [chk4]

CRC16 check sum = [chk1] + [chk2] * 256
Firmware size in bytes = ([chk3] + [chk4] * 256) * 16

The device will calculate a CRC16 over its internal firmware and return the result and the firmware size in quad words (16 byte)

NOTE:
Depending on the actual firmware size the calculation will take some time and the answer will be delayed.

4.37 Header 196 – Request creation date

Transmitted data : <none>
Received data : [date code LSB] [date code MSB]

The date code is stored in a special Money Controls format called RTBY (Relative To Base Year), originally chosen in the mid-Eighties to avoid any Y2K issues.

The returned 16 bits are divided into three parts:
Bit 0..4 = Day 1..31
Bit 5..8 = Month 1..12
Bit 9..13 = Year 0..31 (relative to base year)

4.38 Header 195 – Request last modification date

Transmitted data : <none>
Received data : [date code LSB] [date code MSB]

This command returns the last modification date of the product. It is usually changed by remote programming toolkits or PC-based support software.

The date code formatted is detailed under the 'Request creation date' command.

4.39 Header 194 – Request reject counter

Transmitted data : <none>
Received data : [n1] [n2] [n3]

This command returns the total number of reject coins / bills put through a device.

24-bit counter in decimal = [n1] + 256 * [n2] + 65536 * [n3]
Range 0 to 16,777,215

The counters are stored in EEPROM and their value is retained after a power-down or reset. The counter value is stored every 5 insertions and so should be used as an approximate guide to performance rather than for auditing purposes.

4.40 Header 193 – Request fraud counter

Transmitted data : <none>
Received data : [n1] [n2] [n3]

This command returns the total number of known fraud coins put through the device.

24-bit counter in decimal = [n1] + 256 * [n2] + 65536 * [n3]
Range 0 to 16,777,215

NOTE:
Because the Eagle currently doesn't use the concept of programmed fraud coins this counter will always be zero.

4.41 Header 192 - Request build code

Transmitted data : <none>
Received data : ASCII

The product build code is returned.
With the build code the used ccTalk command set and type of write protection can be identified.

'IT0' = Italian flavor version (ACMII)
'IT1' = Italian flavor version (ACMII), write protected (ACMII)
'DE0' = Default (standard) version
'DE2' = Default (standard) version, write protected (VDAI)

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

4.42 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 will be stored in EEPROM.

4.43 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.

4.44 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

4.44.1 Token

TK1234 = token no. 1234
A token will be characterized by a 4-digit number. No mint issue.

4.44.2 Teachable channels (standard)

XX000A = free channel* used for teach mode (SW504 only)
XX001A = factory prepared teach channel no. 1 (programmed / used)
XX001* = factory prepared teach channel no. 1 (not programmed / used)
XX002* = factory prepared teach channel no. 2 (not programmed / used)

The Coin id of a teachable channel will start with 'XX'. The following 3 digit number reflects the number of the teach channel. Depending on the configuration up to 8 teachable channels are available. If the teach channel wasn't programmed before the last character is an asterisk. After a coin has been learned into this channel the last character will change to an 'A'.

* If a free channel (coin ID = '…..') is used for teaching, every channel will get the same ID after it is programmed

4.44.3 Teachable channels (SR5 compatible)

Option “coin id TEACH” must be activated.
With this option turned on every teachable channel will identify as “TEACH “. The last character is space (20h).

4.45 Header 181 – Modify security setting

Transmitted data : [coin position] [security setting]
Received data : ACK

[coin position]
1 to 16

[security setting]
0 - default setting (nominal performance)
1 to 7 - gradually increase fraud rejection (7 steps)
255 to 249 - gradually increase true acceptance (7 steps)
8 to 248 - undefined

NOTE:
The following security levels are available for the v² eagle

255 = select wide band
0 = select standard band
1 = select narrow band
2 = select super narrow band

If the selected security is not available the nearest available security will be set.
Example: 2,- € coin, standard and narrow band programmed
249..0 will select standard band (0)
1..7 will select narrow band (1)

4.46 Header 180 – Request security setting

Format (a) – request actual level
Transmitted data : [coin position]
Received data : [security setting]

Format (b) – request minimum and maximum level
Transmitted data : [80h | coin position]
Received data : [minimum security] [maximum security]

Example: sent: [03h]
received: [0] ==> coin 3 is currently set to standard band
sent: [83h]
received: [255] [1] ==> coin 3 can be set to wide, standard or narrow

4.47 Header 179 – Modify bank select

Transmitted data : [bank no.]
Received data : ACK

This command is answered for compatibility purpose. No functionality.

4.48 Header 176 – Request alarm counter

Transmitted data : <none>
Received data : [0]

This command is answered for compatibility purpose. No functionality.

4.49 Header 173 – Request thermistor reading

Transmitted data : <none>
Received data : [0]

This command is answered for compatibility purpose. No functionality.

4.50 Header 170 – Request base year

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

The product base year (see base year in ‘Request creation date’ and

'Request last modification date' commands) is returned as a 4 character ASCII string.

e.g. '1999', '2000'.

4.51 Header 169 – Request address mode

Transmitted data : <none>
Received data : [address mode]

This command returns the ccTalk addressing mode to help with automatic re-configuration of ccTalk peripherals. Its use is informational only.

[address mode]
Bit mask :
B0 - Address is stored in Flash / ROM
B1 - Address is stored in RAM
B2 - Address is stored in EEPROM or equivalent
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)

Any bits may be set according to the operating mode of the device.
[0] is returned to indicate another type of addressing mode.

NOTE:
The Eagle has two possible configurations:

01h = address not changeable
84h = address changeable with serial commands and stored non-volatile in EEPROM

The option "address change enable" will turn this feature on.

4.52 Header 165 – Modify variable set

Transmitted data : <variable>
Received data : ACK

This command is answered for compatibility purpose. No functionality.

4.53 Header 162 – Modify inhibit and override registers

Format (a) – 16 coins
Transmitted data : [current : inhibit mask 1] [current : inhibit mask 2]
[current : sorter override bit mask]
[next : inhibit mask 1] [next : inhibit mask 2]
[next : sorter override bit mask]
Received data : ACK

Format (b) – 32 coins
Transmitted data : [current : inhibit mask 1] [current : inhibit mask 2]
[current : sorter override bit mask]
[next : inhibit mask 1] [next : inhibit mask 2]
[next : sorter override bit mask]
[current : inhibit mask 3] [current : inhibit mask 4]
[next : inhibit mask 3] [next : inhibit mask 4]
Received data : ACK

For coin acceptors this command…

- sets inhibits and overrides in one operation (see also the ‘Modify inhibit status’ command and the ‘Modify sorter override status’ command)
- sets both a ‘current’ value and a ‘next coin’ value

The benefit of using this command is that it allows precise coin-by-coin control of the coin acceptor and provides a means of tackling the inherent latency in serial operation. If the next coin is always disabled, the validator will only accept one coin at a time. After each credit (strictly speaking an attempted accept sequence) the host software can request another coin and thus control the overall accept rate. If the next coin overrides are always active, the validator will only route one coin at a time to a payout tube - all other coins will go to the cashbox. The host software can thus maintain a precise fill level in any tube.

NOTE:
This command is implemented for SR5 compatibility and not available on all variants. We recommend using the unbuffered mode instead (refer to header 229).

4.54 Header 150 – Request individual accept counter

Transmitted data : [coin type]
Received data : [0] [0] [0]

This command is answered for compatibility purpose. No functionality.

4.55 Header 141 – Request firmware upgrade capability

Transmitted data : <none>
Received data : [firmware options]

[firmware options]
0 - firmware in ROM / EPROM
1 - firmware in FLASH / EEPROM with upgrade capability

NOTE:
This command is always unencrypted
Depending on variant and configuration this feature is available. Where it is not available the device may not even answer to this command.

0 or no answer = firmware upgrade not supported
1 = firmware upgrade supported

4.56 Header 140 – Upload firmware

Transmitted data : [block no. MSB] [block no. LSB] [data 1] [data 2]… [data 128]
Received data : ACK or NAK

The firmware file must be split into blocks of 128 byte each and transmitted in ascending order to the device. The maximum number is 65536 blocks which gives a total capacity of 256 x 256 x 128 bytes = 8MBytes.

Example:
[2] [130] [0] [140] [0] [0] [….file content (byte 0..127).....] [chk] (block 0)
[2] [130] [0] [140] [0] [1] [….file content (byte 128..255).....] [chk] (block 1)
...
[2] [130] [0] [140] [1] [82] [….file content (byte 43264..43391)..] [chk] (block 338)

NOTE:
This command is always unencrypted

The first block is special purpose. With the information of the first block the bootloader will verify the new firmware.
If the verification fails (e.g. the firmware is not compatible) the answer will be NAK. All subsequent blocks transmitted will be NAKed also
If the verification succeeds the answer is ACK and the bootloader will erase the current firmware and device is ready to receive the next blocks. Because of the erase procedure there will be a delay of up to one second before the ACK is sent.
After the last block is sent the host must use header 138 to finish the upgrade process.

4.57 Header 139 – Begin firmware upgrade

Transmitted data : <none>
Received data : ACK

This command initiates a firmware upgrade. The internal bootloader will be started. It has a reduced instruction set and will identify as “eagle bootloader” if header 244 is requested.

After the ACK is received, the firmware upload can be started sending the first block of the firmware file to the validator.

NOTE:
This command is always unencrypted

Specifics when using encrypted communication:
Depending on the encryption type there are situations (e.g. certain encryption keys) where an encrypted command looks exactly like an unencrypted header 139. In these special cases the Eagle assumes encrypted communication for the first two repetitions. The 3rd repetition will be assumed unencrypted and the firmware upgrade will start.
This means the host should be prepared to send “begin firmware upgrade” up to three times if it doesn't receive an unencrypted ACK as reply.

4.58 Header 138 – Finish firmware upgrade

Transmitted data : <none>
Received data : ACK or NAK

This command terminates a firmware upgrade.
ACK = upgrade successful
NAK = upgrade failed
If the upgrade was successful, the new firmware will be started, otherwise the bootloader will remain active waiting for further commands.

NOTE:
This command is always unencrypted

4.59 Header 111 – Request encryption support

Transmitted data : [170] [85] [0] [0] [85] [170]
Received data : [protocol level] [command level]
[protocol key size]
[command key size] [command block size]
[trusted mode]
[BNV2 | BNV1] [BNV4 | BNV3] [BNV6 | BNV5]
[DES1] [DES2] [DES3] [DES4]
[DES5] [DES6] [DES7] [DES8]

The command description here is a brief outline. For full implementation details with worked examples please refer to the document "ccTalk Serial Protocol Encryption Standard 1.2" or "DES encryption for coin validators"

This command always works unencrypted even if protocol level encryption is in place. Therefore a host machine can always identify the type of encryption being used by a peripheral.

Protocol level encryption is used at the packet level on every single message between the host machine and peripheral. Command level encryption is used to protect the data payload of selected commands only and these vary with the type of peripheral.

[protocol level] 0 – no protocol encryption,
1 – ccTalk Serial Protocol Encryption Standard 1.2

[command level] 0 – no encryption,
101 – DES Encryption

[protocol key size] 0 – no protocol encryption
24 – ccTalk Serial Protocol Encryption Standard 1.2

[command key size] 0 – no command encryption
64 – DES Encryption

[command block size] 0 – no command encryption
64 – DES Encryption

[trusted mode] 0 – normal operating mode
255 – trusted key exchange mode

In trusted key exchange mode, the values of the keys follow; otherwise zero bytes are returned. Entering this mode requires a physical link or equivalent on the product i.e. it cannot be entered remotely through the ccTalk bus for obvious security reasons.

[BNV]	0,0,0 or 3 bytes for the ccTalk encryption algorithm key. It was used originally on banknote validators, hence 'BNV' key.
[DES]	0,0,0,0,0,0,0,0 or 8 bytes for the DES key.

Possible configurations for Eagle:

Unencrypted:	[0].... [0] (17 x)
BNV only:	[1] [0] [24] [0] [0] [0/255] [x] [x] [x] [0] .. [0]
DES only:	[0] [101] [0] [64] [64] [0/255] [0] [0] [0] [x] .. [x]
BNV+DES:	[1] [101] [24] [64] [64] [0/255] [x] [x] [x] [x] .. [x]

x = actual key (trusted mode) or zero (non trusted mode)

4.60 Header 96 – Remote coin programming

Transmitted data : <variable>
Received data : <variable>

With the option "remote coin programming" this command is available and allows coin files to be downloaded into the validator. For a detailed description refer to the "ccTalk remote coin programming" document.

4.61 Header 83 – Disable intelligence

Transmitted data : <none>
Received data : ACK

This command is answered for compatibility purpose. No functionality.

4.62 Header 74 – Calculate configuration checksum

Transmitted data : <none>
Received data : [CRC LSB] [CRC MSB]

This command returns a CRC-16 checksum for the functional (coin independent) settings within the Eagle's internal configuration.

4.63 Header 3 – Clear comms status variables

Transmitted data : <none>
Received data : ACK

This command is answered for compatibility purpose. No functionality.

4.64 Header 2 – Request comms status variables

Transmitted data : <none>
Received data : [0] [0] [0]

This command is answered for compatibility purpose. No functionality.

4.65 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.

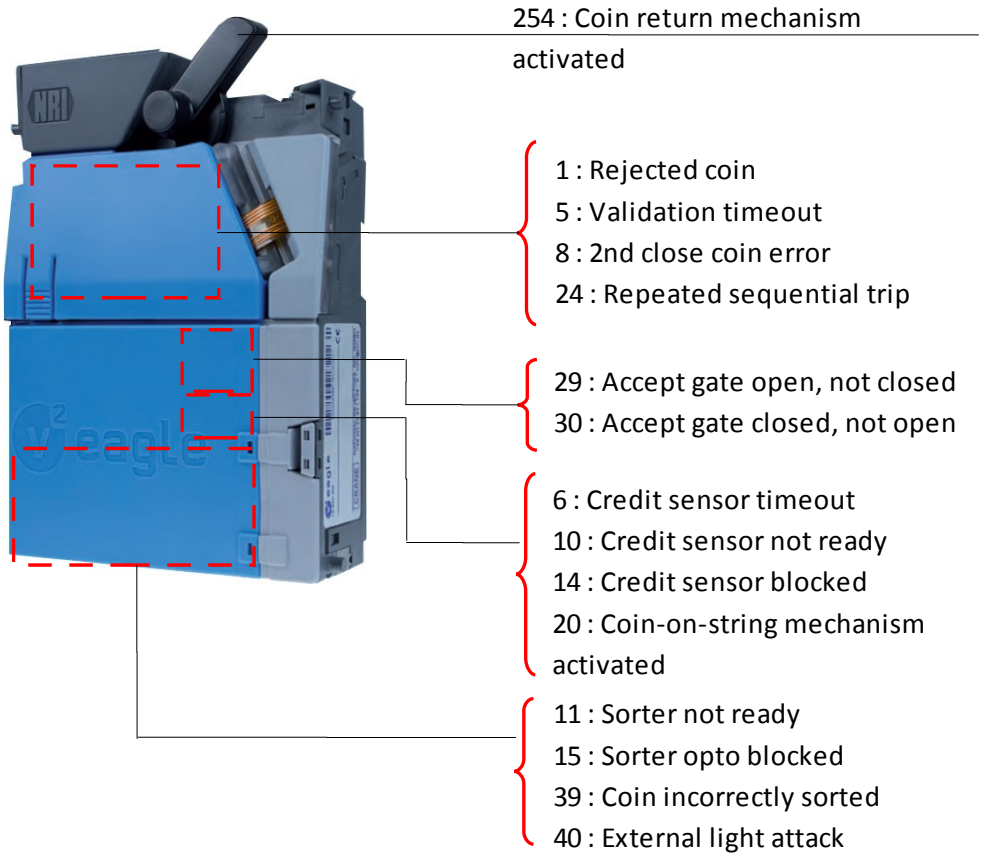
The slave device should return an ACK immediately prior to resetting and allow enough time for the message to be sent back in full.

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

5 Error codes

5.1 Overview

As the coin travels from entry down to the validator exit, several errors can occur. The following picture provides an overview. The full list with description follows in the next section.



5.2 Complete list

code	*ccTalk Error*	*Description*	*Coin rejected*	*Suggested recovering method*
0	Null event (no error)	No error, static error resolved	no	-
1	Rejected coin	Coin was rejected because it was unknown or master inhibit is active	yes	-
5	Validation timeout (dyn/static)	A coin was detected entering the validation area but failed to leave it within the designated time. *Possible coin jam or fraud attempt.* *Validation area became blocked. A Null event will be sent if situation is solved.*	possible	check
6	Credit sensor timeout	Coin reached the credit sensor area but did not leave it within the designated time. *Possible coin jam.* *## SR5: coin didn't reach the post gate credit sensor. Possible coin jam or gate/sensor fault.*	no (+credit) *SR5: possible*	check
8	2nd close coin error	A coin was inserted too close to the one in front. Both coins were rejected.	yes	reduce insertion speed
10	Credit sensor not ready	Coin was rejected because credit sensor is busy detecting previous coin(s)	yes	reduce insertion speed
11	Sorter not ready	A coin was inserted while the sorter flaps for the coin in front were still operating. Coins have been inserted too quickly.	yes	reduce insertion speed
14	Credit sensor blocked (dyn/static error)	Coin was rejected because credit sensor is permanently blocked *Credit sensor is permanently blocked. A Null event will be sent if situation is solved.*	Yes *no*	check
15	Sorter opto blocked (dyn/static error)	Coin was rejected because optical sensor on internal sorter is permanently blocked *One or more optical sensors blocked on internal sorter. A Null event will be sent if situation is solved.*	Yes *no*	check
17	Coin going backwards	A coin was detected going backwards through the coin acceptor. Possible fraud attempt.	no	count
20	C.O.S. mechanism activated (static/dyn error)	A specific sensor for detecting a 'coin on string' was activated. Possible fraud attempt. There will be no credit for this coin.	???	count
21	DCE opto timeout	A coin seen on the optional external coin entry	possible	check

code	*ccTalk Error*	*Description*	*Coin rejected*	*Suggested recovering method*
		sensor was not seen subsequently in the validation area. Possible coin jam.		
22	DCE opto not seen	A coin measured was not seen on the optional external coin entry sensor. Possible fraud attempt.	yes	count
24	Reject coin (repeated sequential trip)	A coin was rejected 5 times in succession with no intervening true coins. Possible fraud attempt.	yes	count
29	Accept gate open not closed	Unknown or inhibited coin was accepted (acceptance gate stuck)	no	stop
30	Accept gate closed not open	Valid coin didn't reach the post gate credit sensor. Possible coin jam, acceptance gate stuck or sensor fault.	possible	observe
31	Manifold opto timeout	A coin was sent into the manifold module (external coin diverter) but was not seen coming out. Possible coin jam.	no (+credit)	check
32	Manifold opto blocked (static/dyn error)	There is a permanent blockage at the manifold module sensor (external coin diverter). The coin acceptor will not accept any more coins. A Null event will be sent if situation is solved.	???	check
39	Coin incorrectly sorted	Diverter failed. Coin was routed to unintended path. The following coin event will report coin and (unintended) path.	no	observe
40	External light attack (static error)	External light is interfering with one of the optical sensors. Possible fraud attempt.	no	count
128 … 159	Inhibited coin (Type 1..32)	A true coin was inserted but was prevented from accepting by the inhibit register.	yes	-
254	Coin return mechanism activated (static error)	An attempt to clear a coin jam by opening the flight deck was detected. The coin acceptor cannot operate until the flight deck is closed. A Null event will be sent if situation is solved.	no	check
255	Unspecified alarm code		no	

Explanation:
(+credit) = credit event can be generated (configuration option)
??? = reject situation is undefined because static and coin driven (dynamic) errors are mixed

5.3 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

stop
Fatal error → set machine out of order

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

check
check static device errors by calling header 232 (perform self-check).
While error is reported → stop coin input.
If the error stays active for more than 5 seconds → reset the Eagle and see if it recovers
If the error is still there → put the machine out of order.

5.4 Event generation options

The following combinable options will influence the way the Eagle is generating events. The options are combined bit wise to a configuration parameter. Option 1/2/3/4 = 01/02/04/08h. All options activated result in 0Fh (= 15 dec.).

Option 1: (01h, credit event after acception faults)
(+credit) in the event code table
The Eagle will credit coins only if they have passed the upper post gate sensor and one of the lower sorter sensors within a specified time. On one hand this is a good protection against manipulation but on the other hand a coin that fails to fulfill the timing constraints for other reason (jam or dirt) will be taken from the customer but not credited. With this option activated a credit event will be reported directly after the error event.

Option 2: (02h, no NULL events for static errors)
If a permanent fault (marked as “static error” in the event code table) is triggering an event the Eagle will report a NULL event after every static error is resolved. This feature can be turned off.

Option 3: (04h, no NULL event after reset)

ccTalk uses a special mechanism to report a device reset to the host: The device will set its event counter to zero and because this is the only time when this is allowed the host can detect this as a reset.
The event counter will be incremented not before a coin is inserted and this could be a problem. From machine start to the first coin insertion the validator is in a state where it is unable to report subsequent resets because the event counter is always zero. Without a cyclic check of the inhibit flags the machine will not be aware of the validator being disabled due to a subsequent reset caused by ESD.

To overcome this weakness the Eagle will add a NULL event to its buffer as soon as the host has read the event buffer for the first time after reset. This NULL event increments the event counter and ensures that the event counter zero is reported only once and therefore every subsequent reset gets noticed by the host.

This mechanism can be switched off.

Option 4: (08h, no events for static errors)
Some event codes will be generated in two situations:

1) if the Eagle changes its error state e.g. a sensor becomes defective (static error)
2) if a coin is rejected because of that error state (coin driven, dynamic error)

Because of this mixture it is not possible to decide if this error happened with or without coin.
Example: Code 15 = sorter opto blocked

- The Eagle will report error 15 if the sensor gets blocked (no coin involved).
- It will generate another code 15 for every coin that is rejected because of this sensor being blocked (coin rejected)

With this option the generation of events for static errors will be suppressed. The 'mixed' event codes 14, 15, 20, 32 will only be coin driven and therefore always imply a rejected coin. They can then be interpreted as 'coin rejected because...'.

NOTE:
To detect if the device is ready to accept coins has to be done by using header 232 (perform self-check)

6 Fault codes

Byte 1	*Byte 2*	*cctalk Error*	*Description*
0	-	no error	No error, static error resolved
1	-	EEPROM checksum corrupted	Configuration data is corrupted *device is damaged*
3	-	Fault on credit sensor	A fault was detected with the post-gate credit sensor. A serial credit can only be generated if the coin passes this sensor *coin jam, dirt, or external light*
6	-	Fault on diameter sensor	A fault was detected on a validation sensor specifically reserved for diameter resolution. *coin jam, dirt or external light*
8	1	Fault on sorter exit sensors	A fault was detected on the sorter exit sensors. These sensors confirm that a coin has cleared the sorter flaps and verify the path taken. *coin jam, dirt or external light*
8	2	Fault on external sorter sensors	A fault was detected on the sensors of the optional external sorter *coin jam or dirt*
19	-	Fault on coin return mechanism	The flight deck is open. Code will be reported as long as customer keeps pressing the return lever. This code is informational and should be ignore until it is present for more than 10 seconds (return mechanism stuck)
20	-	Fault on C.O.S. mechanism	A fault was found on the ‘Coin on String’ sensor. *coin jam, dirt or external light*
30	-	ROM checksum mismatch	Firmware is corrupted *device is damaged*
35	-	D.C.E. fault	Optional coin entry sensor is blocked/damaged/disconnected *coin jam, installation fault*
255	-	Unspecified fault code	A permanent error that is not matching one of the above situations is preventing coin acception

7 Configuration options

The following ordering options have influence on the behavior or the availability of ccTalk commands.

Option	*value*	*Description*
Address change	yes/no	Makes the commands 251 and 250 available. They can be used to change the ccTalk address of the device.
Remote teach	yes/no	Makes the commands 202 and 201 available. They can be used to teach new coins into the device.
Remote coin programming	yes/no	Makes the command 96 available. It can be used to download coin files into the device.
Unbuffered mode	yes/no	The device will no longer buffer up to 5 coin events. Instead it will automatically inhibit after each coin accepted. Refer to section 4.21.4 Unbuffered Mode for details.
Italian flavour command set	yes/no	There are several restrictions for the Italian gaming market that will be enabled by this option. - No answer to command 216/215/214 - using format a for command 209 - build code starts with 'IT'
Coin id 'TEACH'	yes/no	Will report 'TEACH ' for every teachable channel. Refer to section 4.44.3 Teachable channels (SR5 compatible) for details
BNV encryption	yes/no	Will activate the protocol encryption. Refer to the **"ccTalk Serial Protocol Encryption Standard 1.2"** for details
DES encryption	yes/no	Will activate the DES command level encryption. Refer to **"DES encryption for coin validators"** for details.
SR5 compatibility mode	0..15	The Eagle will identify itself as SR5i. The behaviour can be finely graduated by a bit mask that switches this on and off for different headers. Refer to the section 7.1 SR5 compatibility mode for details.
Event generation options	0..15	This value determines how ccTalk events are generated. Refer to section 5.2 Event generation options for details.
32 coin positions	yes/no	With this option activated the Eagle will manage 32 instead of 16 coin positions. All ccTalk commands that take 'coin position' as parameter will accept values 1..32.

7.1 SR5 compatibility mode

ccTalk header	*Data*	*Value (bit mask)*	*0*	*1*
246	Manufacturer ID	Bit 0 (01h)	„NRI“	"Money Controls"
244	Product Code	Bit 1 (02h)	"EAGLE"	"SR5i"
241	software revision	Bit 2 (04h)	„4450100“	"BRM-F1-V1.00"
192	Build Code	Bit 3 (08h)	„DE0“	"STD14 "

8 Operation

8.1 Initial state after reset

Variable	*Value (with NVM)*	*Value (without NVM)*
ccTalk address	= 2 or NVM	= 2*
Default sorter	= NVM	= 1*
Sorter setting (per coin)	= NVM	= 1,1,1,1*
Master inhibit	= 1 (device enabled)	= 1 (device enabled)
Inhibit status	= 0 (all coins disabled)	= 0 (all coins disabled)
Sorter override status	= 255 (normal sorting)	= 255 (normal sorting)

NVM = loaded from non volatile memory
* = these values can be customized, the factory default is given here

8.2 Non volatile memory

If the device is equipped with an NVM (e.g. EEPROM), some values are retained over power-down.

NOTE:
To prevent the NVM from getting worn out, the sorter override mechanism (refer to section 4.32.1 Sorter override mechanism) should be used to override designated paths to cashbox instead of changing the path setting.

NOTE:
Because every device has been tested before shipping, the values read from NVM are undefined at the time being mounted into the machine. Therefore the host should verify the sorter settings during initialization process. It should read the sorter settings first and then send only those settings that have to be changed. This keeps the initialization process short and also prevents the NVM from getting worn out.

8.3 Minimum operation requirements

To operate the v² Eagle 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) every 200ms and process events/errors
4. request header 232 (self check) every second and process fault codes

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

8.4 Delayed activation of settings

If settings (e.g. sorter paths, overrides, inhibits) are changed during operation, they will affect the next coin that is processed. If there are coins already in the system (e.g. within the sorter) the new settings cannot affect them. They will be processed with the old settings.

Therefore host software should be aware of „old state“ coin reports coming from the validator up to 500ms after the settings were changed.

After a coin is inhibited, up to two coins of that type can still be accepted and reported. After a path was changed one or two coins might still be routed to the old path.