

# AL66 Pulse / modifiable ccTalk COIN ACCEPTOR

Operator's Manual

Rev. 2.00



SaxXot Deutschland GmbH, Zeppelinstrasse 73, DE 81669 München  
Tel.: +49 (0)894141446-00 Fax: +49 (089) 4141446-75, info@saxxot.de - www.saxxot.de

**The AL66 coin acceptor can operate by Pulse mode or by modifiable ccTalk protocol.**

It is available in the following versions:

**V** = rejected coin sorted from bottom rear slot, accepted coin sorted from bottom front slot

**I** = rejected coin sorted from bottom front slot, accepted coin sorted from bottom rear slot

**K** = rejected coin sorted from slot low front end, dropping slot on top

**S** = rejected coin sorted from slot low front end, dropping slot at upper front end

**Technical specs**

**Mechanical features**

Format	3½" standard
Size	88 x 102 x 52 mm
Weight	203 g

**Electrical features**

Supply voltage min.	12 V DC (min. 8 V DC)
Supply voltage max.	24 V DC (max. 26 V DC)
Power consumption:	
in acceptance	max. 350 mA(30 ms)/100 mA
in reading	≤ 30 mA
in Stand by	≤ 25 mA
Standard power save	≤ 5 mA
Self-wake up	≤ 6 mA
Output type	Open collector Darlington
Saturation output voltage	≤ 1 V
Output voltage max.	50 V (Active Low)
Output current max.	250 mA
Input trigger voltage min.	3 V (Active High)
Input voltage max	50 V
Input impedance	≈ 55 kΩ

**Coin acceptance**

Coin channels number	16
Coin diametre min.	16 mm
Coin diametre max.	32 mm
Coin thickness	1 to 3,4 mm
Speed	3 coins/sec. (V) – 4 coins/sec. (S, K, I)

**Communication modes**

Pulse	switching by Dip-Switch or by programming software
Modifiable ccTalk	programming software

**Timing data**

Power-up recovery time	≤ 200 ms
Wake-up recovery time	≤ 50 ms
Pulse or time out tolerance	± 2%

**Ambient conditions**

Working ambient temperature	0°C to 60°C
Storage temperature	- 30°C to 70°C
Humidity	up to 75% ( <i>non condensing</i> ) for standard up to 95% for tropicalized version

**EMC performance**

This product is compliant with EN55014-1 and EN55014-2 test specification

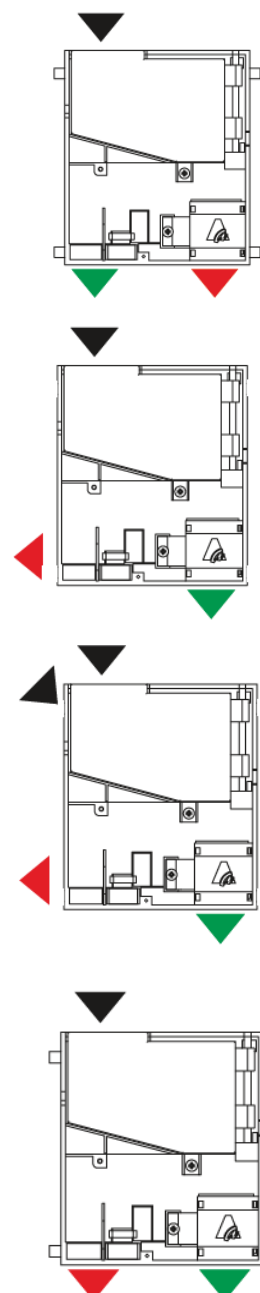
> 8-bit top-performing microcontroller with 36 KB Flash-memory, highly immune to magnetic interference and to environmental conditions.

> Six magnetic sensors and one optic detector combine to guarantee the most accurate capacity of discrimination. Data-digitizer interface designed to cope at best with bi-metal and magnetized coins. The system sharpens selectivity and security, and makes the programming process easy and faster.

> One wire-shearer and one wire-tearer in version V provide mechanical shielding against fishing-fraud. Coin-Guard anti-cheat system, based on the synergy of 3 suitably positioned optic sensors.



- INGRESSO MONETA COIN ENTRY
- MONETA VALIDA VALID COIN
- MONETA RIFIUTATA REJECTED COIN



## Connections

The selector connects to other peripherals and to the main board by the following sockets:

**X1. Power supply and standard Pulse interface.** Connector X1 is an IDC 10-p socket, whose pin-out is shown in grid aside. It consists of:

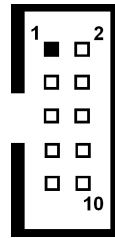
- 2 power supply pins (*pin 1 common/ground & pin 2 positive*), 6 “open collector” outputs (*pin 3,4,7,8,9 & 10*),

**ACTIVE STATE = LOW**

- 1 entry pin (*pin 6, normally used to inhibit acceptance of coins*),

**ACTIVE STATE = HIGH**

- 1 double function pin (*pin 5*), either the “open collector” output normally dedicated to the accu multi-pulse, or additional input (i.e. for credit request).

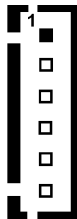


nr.	Description
1	Gnd
2	8-26 Vdc
3	Out 5 / sorter coil B
4	Out 6 / sorter coil A
5	Out 7 (totalizer) / In 2
6	In 1 (inhibit)
7	Out 1
8	Out 2
9	Out 3
10	Out 4 / sorter coil C

Upon request, the acceptor can be prepared for reverse supply polarity, so as to emulate spanish type coin acceptors.

**X2. Display.** NOT AVAILABLE IN THIS HARDWARE VERSION.

Socket X2 (6 pins) is meant for display connection. Displays with SPI or I<sup>2</sup>C bus protocols are supported (see aside).



nr.	Description
1	5 Vdc
2	Gnd
3	12 Vdc
4	Data
5	Data
6	Data

The acceptor must be preset for the display by the manufacturer, then it can be modified by the Alberici dedicated software.

The following models are supported:

- MC 14499 4-digits
- MC 14489 5-digit
- MAX 7219 6-digits
- M643 8-digit LCD
- AD01 2 x 16 digits

**X3. CCTALK.** The X3 4-p socket is for serial **ccTalk®** communication with the main board. The protocol is preset for “slave mode” operation, and is described in AL66 technical manual.

This socket can be used on standard selectors and modifiable cctalk selectors to change the programme of the selector by PC and Alberici free software.



nr.	Descrizione
1	Dati
2	Gnd
3	NC
4	12 Vdc

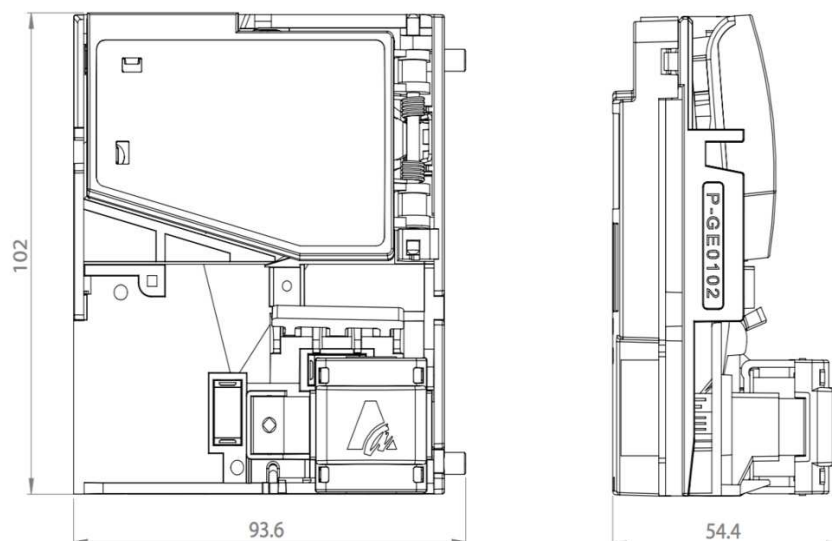
**NOTICE:** The coin selector can acknowledge 16 different coin pieces; one for each of the 16 available channels.

Coins can also be manually programmed by the 2 Dip-Switch Rows SW1 and SW2 located on the acceptor rear side (see below ‘AL66 AUTO-PROGRAMMING INSTRUCTIONS’).

**TAKE CARE!** The coin selector must be installed 90°-95° with respect to level plane. Because of the built-in advanced security system, it is essential that the whole coin path gets not hindered.

Alberici is not responsible for any malfunctioning due to lack of compliance with such recommendations.

## Dimensions



**DISABLE PROGRAMMED COINS**

<p>TO INHIBIT ANY COINS, MOVE TO POSITION ON THE DIP-SWITCH CORRESPONDING TO THE CHANNEL IN WHICH THE COIN IS PROGRAMMED (see grid aside). THEN SWITCH OFF AND ON AGAIN.</p> <p>TO FIND THE CHANNEL THAT CONTAINS THE COIN TO BE DISABLED, PLEASE CHECK THE "CH" COLUMN IN THE SELECTOR LABEL GRID.</p>	dip-switch nr. in row	Channel nr.
	SW1	
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6

**SET THE ACCEPTANCE TOLERANCE LEVEL**

<p>THE DISCRIMINATION CAPACITY OF THE ACCEPTOR CAN BE INCREASED BY MOVING THE DIP-SWITCH 6 IN ROW SW2 TO POSITION 'ON'.</p>	Position of dip-switch 6 (row SW2)	Discrimination capacity
	ON OFF	HIGH discrimination LOW discrimination

**SETTINGS by SW1 ROW**

SW1 Dip-Switch Row	DS1	DS2	DS3	DS4	DS5	DS6
ON	Enable CH1 (ex. 2€)	Enable CH2 (ex. 1€)	Enable CH3 (ex. 0,50€)	Enable CH4 (ex. 0,20€)	Enable CH5 (ex. 0,10€)	Enable CH6 (ex. 0,05€)
OFF	Disable CH1 (ex. 2€)	Disable CH2 (ex. 1€)	Disable CH3 (ex. 0,50€)	Disable CH4 (ex. 0,20€)	Disable CH5 (ex. 0,10€)	Disable CH6 (ex. 0,05€)

**SETTINGS by SW2 ROW**

SW2 Dip-Switch Row	DS2 OFF	DS2 ON	DS4 OFF	DS4 ON	
DS1 OFF	PULSE	MDB (*)			
DS1 ON	CCTALK	SAS (*)			
DS3 OFF					Multi-pulse totalizer, OUTPUT as per programmed setting
DS3 ON					Multi-pulse totalizer on OUTPUT 3
DS5 OFF	Modify credit value		x 1	x 4	
DS5 ON			x 2	: 10	
DS6 OFF					STANDARD Discrimination (L / SLC)
DS6 ON					HIGH Discrimination (H / SLC)

(\*) Not available in this version

## AL66 AUTO-PROGRAMMING INSTRUCTIONS

### A) HOW TO PROGRAMME (NEW) COINS


- 1) Take note of the positions of the dip-switches.
- 2) Power off the coin acceptor.
- 3) Set all the dip-switches in SW1 row to OFF.
- 4) Move the dip-switch 2 of the SW2 row to ON.
- 5) Power up the coin acceptor: start of the auto-programming mode is confirmed by single coil activation.
- 6) In SW1 row, move to ON the dip-switch corresponding to the channel in which the coins will be teached.
- 7) Drop 10 such coins into the acceptor: double coil activation will confirm that the coins have been programmed.
- 8) Power off the coin acceptor and set the dip-switches according to the desired operation.

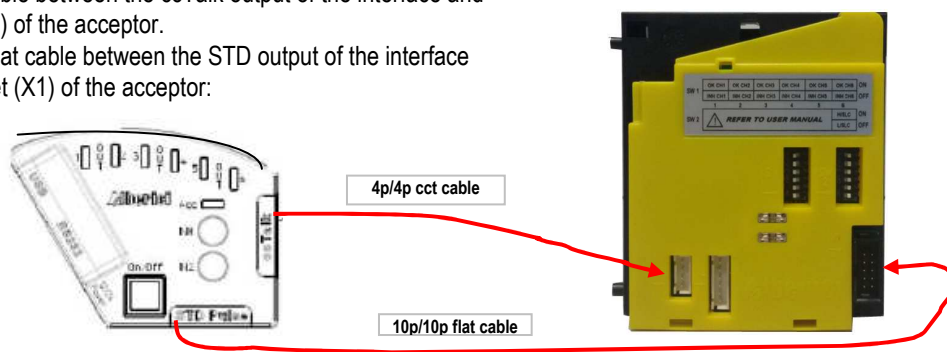
### B) HOW TO RESET the Auto-programmed configurations

- 1) Power off the coin acceptor.
- 2) Set all the dip-switches in SW1 row to OFF.
- 3) Move the dip-switch 2 of the SW2 row to ON.
- 4) Set to ON both dip-switches 1 and 6 of the SW1 row.
- 5) Power up the coin acceptor: after few seconds, triple coil activation will confirm that the coins/channels that have been programmed manually have been cancelled, and that the previous configuration as been restored.
- 6) Power off the coin acceptor and set the dip-switches according to the desired operation.

## PROGRAMMING THE AL66 PULSE / MOD. CCT BY PC SOFTWARE

**THIS IS DONE THROUGH THE K-P1C-000009 PROGRAMMING KIT AND THE "Programming sw 66 v. 2 - Alberici coin selector" SOFTWARE. THE SOFTWARE ITSELF AND ITS MANUAL, CONTAINING INSTRUCTIONS FOR KIT CONNECTIONS AND PROGRAMMING PROCEDURES, ARE BOTH FREELY AVAILABLE FOR DOWNLOAD ON OUR WEB SITE [www.alberici.net](http://www.alberici.net).**

- 1 Download the package including the drivers for the programming interface tool and the application "Programming sw 66 v. 2 - AlbericiCoinSelector.exe" (from our web site "Downloads" section).
- 2 Install the drivers into your PC, then install the application; its icon will be shown on your PC Desktop: 
- 3 Make use of the programming-testing-power tool K-P1C-000009.
- 4 Make sure that voltage supply is off.
- 5 Connect the USB port of the PC to the USB A port of the interface.
- 6 Connect the 4pin cable between the ccTalk output of the interface and the 4-pin socket (X3) of the acceptor.
- 7 Connect the 10pin flat cable between the STD output of the interface and the 10pin socket (X1) of the acceptor:



- 8 Power the selector (green pushbutton on the interface tool).
- 9 Launch the application and modify the acceptor's data appearing on the screen according to your needs. The software interface is very simple and user-friendly; in case, its Instruction Manual is available on the same page of our web site.
- 10 Download the new configuration to the coin acceptor, switch power supply to selector off and then on again.
- 11 Test that operation corresponds to the new configuration programmed.
- 12 Switch power off and disconnect the selector from the PC USB port.

## AL66 MOD. CCT HEADERS

New generation of coin selectors AL55 or AL66 use **cctalk®** communication protocol. This protocol was developed by company Emulator M(ex. Coin Controls) to enable connection of different peripheral devices<sup>1</sup> in small network. Protocol is mostly used in gaming and casino machines but it can be implemented in any other types of machines that use same type of devices. It is public protocol and free to use.

### The communication protocol of ALBERICI coin selectors AL55/66 comply to generic specification 4.4

The following ccTalk commands are implemented for the serial operation of the coin acceptor:

254-Simple poll	227- Request master inhibit status
253-Address poll (broadcast)	226- Request insertion counter
252-Address clash (no bcast)	225- Request acceptance counter
252-Address clash (broadcast)	210- Modify sorter paths
249- Request polling priority	209- Request sorter paths
248- Request status	197- Calculate ROM checksum
246- Request manufacturer id	196- Request creation date
245- Request equipment category id	195- Request last modification date
244- Request product code	194- Request reject counter
242- Request serial number	193- Request fraud counter
241- Request software revision	192- Request build code
240- Test solenoids	188- Request default sorter path
237- Read input lines	184- Request coin id
236- Read opto states	170- Request base year
232- Perform self test	4- Request comms revision
231- Modify inhibit status	3- Clear comms status variables
230- Request inhibit status	2- Request comms status variables
229- Read buffered credit or error codes	1- Reset device
228- Modify master inhibit status	

The following pages show detailed information and examples of the communication messages.

### 6.1 Communication specifications

Serial communication was derivated from RS232 standard.

It is low data rate NRZ (*Non Return to Zero*) asynchronous communication with:

Baud rate 9600, 1 start bit, 8 data bits, no parity, 1 stop bit.

RS232 handshaking signals (*RTS, CTS, DTR, DCD, DSR*) are not supported.

Message integrity is controlled by means of checksum calculation.

#### 6.1.1 Baud rate

The baud rate of 9600 was chosen as compromise between cost and speed.

Timing tolerances is same as in RS232 protocol and it should be less than 4%.

#### 6.1.2 Voltage level

To reduce the costs of connections the "Level shifted" version of RS232 is used.

The idle state on serial connector is 5V, and active state is 0V.

Mark state (*idle*)                    +5V nominal                    from 3.5V to 5V

Space state (*active*)                0V nominal                        from 0.0V to 1.0V

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

<sup>1</sup>Coin selectors, Hoppers(*pay out device*), Banknote readers etc.

### 6.1.3 Connection

The connection of Coin selector at network is achieved by means of 4 pole JST connector (*standard type 7*). Connector is used for power supply and communication as well. For schematics and and connector appearance see image1.

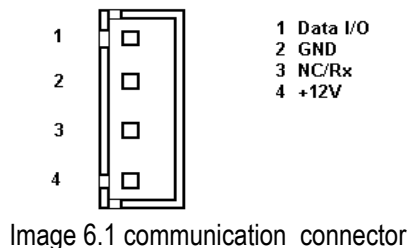


Image 6.1 communication connector

**Recommended periferal connector is:  
JST B 4B-XH-A with crimping contacts SXH-001T-P0.6**

### 6.2 Message structure

Each communication sequence consists of two message string. Message string in case of simple checksum use is structured as follows:

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

There is an exeption of mesage structure when device respond to instruction Address poll and Address clash<sup>2</sup>. The respond consists of only one byte representing address delayed for time proportional to address value. For CRC checksum case format is:

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

#### 6.2.1 Address

Address range is from address 0 to address 255. Address 0 is special case or so caled "brodcast" address and address 1 is default host address.

Table 6.1 shows the recommended address values of different devices.

Device category	Address	Add. addr.	Note
Coin Acceptor	2	11 - 17	Coin validator, selector,
Payout	3	4 - 10	Hopper
Reel	30	31 - 34	
Bill validator	40	41 - 47	Banknote reader
Card Reader	50		-
Display	60		Alphanumeric LC display
Keypad	70		-
Dongle	80	85 - 89	Safety equipment
Meter	90		Replacement for el.mec. counters
Power	100		Power supply
Printer	110		Ticket printing
RNG	120		Random Number Generator

Table 6.1 Standard address for different types of devices

Address for ALBERICI coin selectors AL55/66 is factory set at value 2.

User can change the default address using MDCES instructions: **Address change** or **Address random**.

<sup>2</sup> For details see cctalk44-2.pdf, Address poll

### 6.2.2 Number of data byte

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

Value 0 means that there are no data bytes in the message, and total length of message string will be 5 bytes. Although theoretically it will be possible to send 255 bytes of data because of some limitations in small micro controllers the number is limited to 252<sup>3</sup>.

### 6.2.3 Command headers (*Instructions*)

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

**Header 0** stands for **ACK** (*acknowledge*) replay of device to host.

**Header 5** stands for **NAK** (*No acknowledge*) replay of device to host.

**Header 6** is **BUSY** replay of device to host.

In all three cases no data bytes are transferred. Use of ACK and NAK headers are explained later on, for each specific message transfer.

Commands are divided into several groups according to application specifics:

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

### 6.2.4 Data

There is no limitation in use of data formats. Data could be BCD (*Binary Coded Decimal*) numbers, Hexa numbers or ASCII strings. Interpretation as well as format is specific to each header use, and will be explained in separate chapter.

### 6.2.5 Checksum

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

Simple checksum is made by 8 bit addition (modulus 256) of all the bytes in the message. If message is received and the addition of all bytes are non-zero then an error has occurred<sup>4</sup>.

For noisy environment or higher security application it is possible to use more complex, 16 bit CRC CCITT checksum based on a polynomial of:

$x^{16} + x^{12} + x^5 + 1$  and initial value of CRC register **0x0000**.

Coin selectors AL55/66 use simple checksum, but they can be set to operate by CRC-16 checksum on customer demand.

## 6.3 Timing specification

The timing requirements of cctalk are not very critical but there are some important recommendations.

### 6.3.1 Time between two bytes

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

### 6.3.2 Time between command and reply

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

Other commands that must activate some actions in device, may return reply after the action is finished<sup>5</sup>.

### 6.3.3 Start-up time

After the power-up sequence coin selector should be ready to accept and respond to a cctalk message within time period of **less than 250 ms**.

During that period all internal check-up and system settings must be done, and coin acceptor should be able to recognize and accept coins.

---

<sup>3</sup> 252 bytes of data, source address, header and checksum (total of 255 bytes)

<sup>4</sup> See Error handling

<sup>5</sup> I.e. more than 100 milli sec for solenoid testing.



## 6.4 Error handling

If slave device receive the message with bad checksum or missing data no further action is taken and receive buffer will be cleared.

Host software should decide to re-transmit message immediately or after a fixed amount of time. In case when host receive message with error it has same options.

## 6.5 Command headers

Command header set, that host could use in communication with coin selectors AL55 and AL66 is given in the table 6.2.

Command headers are divided in to 3 different groups:

- Common command headers
- Coin acceptor command headers
- MDCES command headers

Code	Command header	Note
254	FE	Simple poll
253	FD	Address poll (broadcast)
252	FC	Address clash
249	F9	Request polling priority
248	F8	Request status
246	F6	Request manufacturer id
245	F5	Request equipment category id
244	F4	Request product code
242	F2	Request serial number
241	F1	Request software revision
240	F0	Test solenoids
237	ED	Read input lines
236	EC	Read opto states
232	E8	Perform self test
231	E7	Modify inhibit status
230	E6	Request inhibit status
229	E5	Read buffered credit or error codes
228	E4	Modify master inhibit status
227	E3	Request master inhibit status
226	E2	Request insertion counter
225	E1	Request acceptance counter
210	D2	Modify sorter paths
209	D1	Request sorter paths
197	C5	Calculate ROM checksum
196	C4	Request creation date
195	C3	Request last modification date
194	C2	Request reject counter
193	C1	Request fraud counter
192	C0	Request build code
188	BC	Request default sorter path
184	B8	Request coin id
170	AA	Request base year
4	04	Request comms revision
3	03	Clear comms status variables
2	02	Request comms status variables
1	01	Reset device

Table 6.2 cctalk instruction header list

### 6.5.1 Common command headers

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

#### 6.5.1.1 Command 254 [hexFE], Simple poll

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

Addressed device-coin selector respond with ACK (*Acknowledge*).

If within predicted amount of time Coin selector does not respond coin selector is probably not connected, powered or simple not working properly.

Message format is:

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

Coin s. respond: [01] [00] [Dir] [00] [Chk]

As coin selector default address is 2, example of message string is:

Host sends: [02] [00] [01] [FE] [FF]

Coin s. respond: [01] [00] [02] [00] [FD] ACK message

---

#### 6.5.1.2 Command 246 [hexF6], Request manufacturer ID

Coin selector respond with ASCII string representing manufacturer name.

Message format is:

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

Coin s. respond: [01] [Nr.b] [Dir] [00] [a1] [a2] . . . [an] [Chk]

Nr. b is number of data bytes-characters sent by coin selector, and a1 to an are ASCII characters. For **Alberici** coin selector example of message string is:

Host sends: [02] [00] [01] [F6] [07]

Coin s. respond: [01] [08] [02] [00] [41][6C][62][65][72][69][63][69] [DA]

#### 6.5.1.3 Command 245 [hexF5], Request equipment category ID

Respond to command header is standardized name for coin selectors, coin validators or coin mechs. Coin selector respond with ASCII string of characters representing standardized name for that type of device **Coin Acceptor**.

Message format is:

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

Coin s. respond: [01] [0D] [Dir] [00] [43][6F][69][6E][20][41][63][63][65][70][74][6F][72] [Chk]

Number of data byte is always 13, hex [0D].

Example of message string for coin selector(*address 2*) is:

Host sends: [02] [00] [01] [F5] [08]

Coin s. respond: [01] [0D] [02] [00] [43][6F][69][6E][20][41][63][63][65][70][74][6F][72] [16]

#### 6.5.1.4 Command 244 [hexF4], Request product code

Coin selector respond with ASCII string of character, representing the factory type of coin selector. For ALBERICI coin selectors of new generation possible response will be:

- AL55V1, AL55K1, AL55I1

- AL66V2, AL66K3, AL66I3

In special version for italian gambling machines response is allways **AL05V-c** .

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

Coin s. respond: [01] [07] [Dir] [00] [a1][a2] . . . [a7] [Chk]

Number of data bytes sent by coin selector is 6 or 7, hex [07].

Example of message string for coin selector(*address 2*) type **AL06V-c** is:

Host sends: [02] [00] [01] [F4] [09]

Coin s. respond: [01] [07] [02] [00] [41][4C][30][36][56][2D][63] [1D]

#### 6.5.1.5 Command 242 [hexF2], Request serial number

Coin selector respond with three byte serial number. Message format is:

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

Coin s. respond: [01] [03] [Dir] [00] [Serial 1 - LSB] [Serial 2] [Serial 3 - MSB] [Chk]

Serial 1 – first data byte sent is LSB of serial number.

Example of message string for coin selector(address 2) with serial number: 1234567 (hex [BC][61][4E]) is:

Host sends: [02] [00] [01] [F2] [0B]

Coin s. respond: [01] [03] [02] [00] [4E][61][BC] [8F]

#### 6.5.1.6 Command 241 [hexF1], Request software revision

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

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

Coin s. respond: [01] [Nr.b] [Dir] [00] [a1] [a2].... [an] [Chk]

Number of data bytes in ASCII string is not limited and each producer has it's own system of labelling. Example of message string for coin selector(address 2) is:

Host sends: [02] [00] [01] [F1] [0C]

Coin s. respond: [01] [09] [02] [00] [75][31][2E][30][20][70][31][2E][30][2E][30] [71]

Coin selector respond is 'u1.0 p1.0.0'.

ALBERICI coin selectors has program firmware label divided in two parts.

First label u is for protected FLASH memory program(*monitor program*) revision.

First digit is for major changes and second for minor changes. In this case it is u1.0.

Second label is revision of main program FLASH memory.

Main program software revision labelling use 3 digits. First most significant digit is for major software changes, second is for minor software changes and third for "bug" correction. In this case it is u1.0.0.

.....

#### 6.5.1.7 Command 197 [hexC5], Calculate ROM checksum

Coin selector respond with four bytes of micro controller internal memory checksum. First two bytes are program ROM CRC and the second is data EEPROM CRC. Any changes in program or data will change the respond of coin selector.

Message format is:

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

Coin s. respond: [01] [4] [Dir] [00] [CRC1-H][CRC1-L] [CRC2-H] [CRC2-L] [Chk]

Example of message string for coin selector(address 2) is:

Host sends: [02] [00] [01] [C5] [38]

Coin s. respond: [01] [04] [02] [00] [D9][2A][7E][79] [96]

.....

#### 6.5.1.8 Command 192 [hexC0], Request build code

Coin selector respond with ASCII string of character representing it's hardware version and revision. Last revision of printed circuit board for coin selectors AL55/66 is:

**AL66 V1.0.** Message format is:

Host sends: [Dir] [00] [01] [C0] [Chk]

Coin s. respond: [01] [Nr.b] [Dir] [00] [a1] [a2].... [an] [Chk]

Example of message string for coin selector(address 2) is:

Host sends: [02] [00] [01] [C0] [3D]

Coin s. respond: [01] [09] [02] [00] [41][4C][2D][30][35][20][56][35][30] [FA]

.....

#### 6.5.1.10 Command 4 [hex04], Request comms revision

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

Host sends: [Dir] [00] [01] [04] [Chk]

Coin s. respond: [01] [03] [Dir] [00] [Level] [Mag.rev.] [min. rev.] [Chk]

Example of message string for coin selector(address 2) with level of implementation 1, cctalk protocol issue 4.4 is:

Host sends: [02] [00] [01] [04] [F9]

Coin s. respond: [01] [03] [02] [00] [01][04][04] [F1]

#### 6.5.1.11 Command 3 [hex03], Clear comms status variables

After acceptance of command header 3, coin selector clears all three bytes of communication errors counters and respond with ACK message. Message format is:

Host sends: [Dir] [00] [01] [03] [Chk]

Coin s. respond: [01] [00] [Dir] [00] [Chk] ACK message

Example of message string for coin selector(address 2) is:

Host sends: [02] [00] [01] [03] [FA]

Coin s. respond: [01] [00] [02] [00] [FD] ACK message

#### 6.5.1.12 Command 2 [hex02], Request comms status variables

Coin selector respond with three byte data representing communication errors.

First byte is receive time out counter, second byte is number of ignored receive bytes<sup>6</sup> and third byte is number of checksum errors. Message format is:

Host sends: [Dir] [00] [01] [02] [Chk]

Coin s. respond: [01] [03] [Dir] [RxErr1] [RxErr2] [RxErr3] [Chk]

Example of message string for coin selector(address 2) with no errors is:

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

Coin s. respond: [01] [03] [02] [00] [00] [00] [00] [FA]

#### 6.5.1.13 Command 1 [hex01], Reset device

After acceptance of Reset command, coin selector execute software reset and clear all variables in RAM or set them at default value, including different counters and credit buffer. ACK message is sent before reset of coin selector. Host software must set again:

- inhibit state
- sorter path
- master inhibit (*if necessary*)

Message format is:

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

Coin s. respond: [01] [00] [Dir] [00] [Chk] ACK message

Example of message string for coin selector(address 2) is:

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

Coin s. respond: [01] [00] [02] [00] [FD] ACK message

Host software must wait at least **100 ms**, to continue communication with coin selector after reset instruction!

### 6.5.2 Coin acceptor specific command headers

Coin selectors use some specific commands, mostly for control of coin input, acceptance and direction<sup>7</sup>.

Some commands are shared with other device like banknote reader or payout device.

<sup>6</sup> Number of receive buffer overflow bytes.

<sup>7</sup> Sorter control commands

### 6.5.2.1 Command 249 [hexF9], Request polling priority

Basic principle of detecting credit input or eventual errors from coin selector is sequential polling<sup>8</sup>. Coin selectors due to differences in mechanical and electrical construction has different acceptance speed. All events are registered in memory buffer with limited size<sup>9</sup>. To avoid credit loss, host must read coin selector credit buffer within limited time period. Coin selector has internal mechanism to block the coin acceptance and registration of all events if polling time elapse.

For ALBERICI coin selector acceptance speed is from 3 to 4 coins per second<sup>10</sup>.

Considering that it is possible to register 5 event in the buffer, the adequate polling time will be about 1 sec. Because of necessity to register even "close" and non accepted coins polling time must be even shorter.

For ALBERICI coin selectors AL55/66 using cctalk interface, poll time is set to 500 ms.

Coin selectors that use standard 10 pole interface are not necessary to poll.

In that case polling time unit is set to 0(*no polling*)!

Minimum time for polling must not be shorter than overall message time<sup>11</sup>.

Coin selector respond to command with two bytes of data. First byte is poll time unit and second is polling time value<sup>12</sup>.

Message format is:

Host sends: [Dir] [00] [01] [F9] [Chk]

Coin s. respond: [01] [01] [Dir] [Time] [Chk]

Example of message string for coin selector(*address 2*) is:

Host sends: [02] [00] [01] [F9] [04]

Coin s. respond: [01] [02] [02] [00] [02] [32] [C7]

First byte **02** is unit **x10ms**, and second byte is time value **hex32 = 50**.

Polling time is calculated as:

$$T = 10 \times 50 = 500 \text{ ms}$$

### 6.5.2.2 Command 248 [hexF8], Request Status

ALBERICI coin selectors has no additional COS<sup>13</sup> and return mechanism.

Response to that command is always hex[00], coin selector Ok.

Example of message string for coin selector(*address 2*) is:

Host sends: [02] [00] [01] [F8] [05]

Coin s. respond: [01] [01] [02] [00] [00] [FC]

### 6.5.2.4 Command 240 [hexF0], Test solenoids

Host sends one byte mask to determinate which solenoid must be tested.

Coin selector accept gate solenoid or sorter solenoid will be switched on for period of 100 ms and after that, ACK message will be transmitted. Message format is:

Host sends: [Dir] [01] [01] [F0] [Mask.] [Chk]

Coin s. respond: [01] [00] [Dir] [00] [Chk] ACK

Example of message string for coin selector(*address 2*) acceptance gate test is:

Host sends: [02] [01] [01] [F0] [01] [0B]

Coin s. respond: [01] [00] [02] [00] [FD] Single click -> 100 ms, ACK

Bit position for output that is used to drive sorter coil are:

bit 0 = accept gate coil

bit 1 = sorter coil "A"(out 6/pin 4)

bit 2 = sorter coil "B"(out 5/pin 3)

bit 3 = sorter coil "C"(out 4/pin 10)

If output selected with bit in mask is not programmed for sorter activation it will not be activated but coin selector will still response with ACK.

<sup>8</sup> Reading memory buffer from coin selector

<sup>9</sup> Five stage double byte memory buffer

<sup>10</sup> Dependant on mechanical type of coin selector (K, S type is faster ) and coin

<sup>11</sup> For coin selector with respond time 2 ms and byte gap 1 ms it is 38 ms

<sup>12</sup> For details see, cctalk44-2.pdf

<sup>13</sup> Coin On String

### 6.5.2.6 Command 237 [hexED], Read input lines

Coin selector respond with two data byte representing state of DIP-switches and state of inputs In1(*pin 6*) and In2(*pin 5*)<sup>14</sup>. ALBERICI coin selectors has one or two banks of DIP-switches for various data or operating modes setting. First data byte is state of first DIP-switch(*bank 1*) and In1, while second represent second DIP-switch(*bank 2*) and In2. LSb is first switch in bank and MSb is state of input. Switch closed state is represented with logic "1", and input active state is logic "1". Message format is:

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

Coin s. respond: [01] [02] [Dir] [Mask1] [Mask2] [Chk]

Example of message string for coin selector(*address 2*), with all switches "off" and inputs not active is:

Host sends: [02] [00] [01] [ED] [10]

Coin s. respond: [01] [02] [02] [00] [00] [00] [FB]

Example of message string for coin selector(*address 2*), with all switches "on" and input 1(*inhibit acceptance*) active is:

Host sends: [02] [00] [01] [ED] [10]

Coin s. respond: [01] [02] [02] [00] [BF] [00] [3C]

### 6.5.2.7 Command 236 [hexEC], Read opto states

Coin selector respond with one data byte representing the state of opto pairs.

ALBERICI coin selectors has up to 3 pairs of optical sensor<sup>15</sup> for detection of coin position, speed and direction and 2 pairs of opto sensors for diameter measurement.

Bit position for opto pairs are:

- bit 0 Diam. measure opto 1
- bit 1 Diam. measure opto 2
- bit 2 Control opto 1
- bit 3 Control opto 2
- bit 4 Control opto 3
- bit 5 Not used
- bit 6 Not used
- bit 7 Not used

Control opto sensor 2 is called "credit" opto sensor exist in all version of coin selectors and it is placed after the acceptance gate. Other pairs are optional and some coin selectors has 2 and some 3 control optical pairs. Number of control pairs make part of coin selector type label. For example coin selector type AL66V2 has 2 control opto sensor pairs. The unused bits or non existing optical sensors are always read as 0.

Interruption of light barrier of opto sensor correspond to bit value 1.

Message format is:

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

Coin s. respond: [01] [01] [Dir] [Mask.] [Chk]

Example of message string for coin selector(*address 2*) with opto sensors cleared is:

Host sends: [02] [00] [01] [EC] [11]

Coin s. respond: [01] [01] [02] [00] [00] [FC]

### 6.5.2.9 Command 232 [hexE8], Perform self-test

Coin selector respond to command with one or two bytes of data according to

table 6.3. First byte is fault code and second is optional data, usually representing fault sensor number(*from 1 to 3*).

Code	Fault	Optional data	Comment
0	OK No fault detected	-	-
2	Fault on inductive sensor	Sensor number	-
3	Fault on credit sensor	-	Control opto sensor 2
6	Fault on diameter sensor	-	-
18	Fault on reject sensor	-	Control opto sensor 3
33	Power supply out of limits	-	-
34	Temperature out of limit	-	Optional
255	Unspecified fault code	-	-

Table 6.3 Fault codes for AL55/66 coin selectors

<sup>14</sup> If In2 is programmed as input

<sup>15</sup> In some case group could contain more than one opto pairs

Inductive sensor numbers are:

01 Upper inductive sensor

02 First lower inductive sensor

03 Second lower inductive sensor

Message format is:

Host sends: [Dir] [00] [01] [E8] [Chk]

Coin s. respond: [01] [01/02] [Dir] [Fault c.][Data opt.] [Chk]

Example of message string for coin selector(*address 2*) with no fault detected is:

Host sends: [02] [00] [01] [E8] [15]

Coin s. respond: [01] [01] [02] [00] [00] [FC] No fault detected

Example of message string for coin selector(*address 2*) with first lower sensor fault detected is:

Host sends: [02] [00] [01] [E8] [15]

Coin s. respond: [01] [02] [02] [00] [02][02] [F7] Fault on first lower sensor detected

#### 6.5.2.10 Command 231 [hexE7], Modify inhibit status

With this command host is able to inhibit the acceptance of some or all coins.

Acceptance or inhibition is set with a two byte mask sent by host.

Bits from 0 to 15 determinate coin positions from 1 to 16<sup>16</sup>.

Number of coin channels in new ALBERICI coin selectors AL55/66 is same as number of position(16). Message format is:

Host sends: [Dir] [02] [01] [E7] [LSB Mask.] [MSB Mask.] [Chk]

Coin s. respond: [01] [00] [Dir] [00] [Chk] ACK

Example of message string to enable all position for coin selector(*address 2*) is:

Host sends: [02] [02] [01] [E7] [FF] [FF] [16]

Coin s. respond: [01] [00] [02] [00] [FD] ACK

After that all programmed coins will be enabled. Command has no effect on coin position that are not programmed.

Initially coin channels could be programmed with acceptance enabled or disabled.

**For coin selectors that are using only cctalk interface, all coins position must be initially inhibited!**

#### 6.5.2.11 Command 230 [hexE6], Request inhibit status

Coin selector respond with two byte data that correspond to inhibit state mask for all 16 positions of coin. If bit value is 1 acceptance of coin in that position is enabled. If bit value is 0 coin is inhibited. Message format is:

Host sends: [Dir] [02] [00] [E6] [Chk]

Coin s. respond: [01] [02] [Dir] [00] [LSB Mask.] [MSB Mask.] [Chk]

Example of message string for coin selector(*address 2*) AL06V-c<sup>17</sup> after power-up or reset is:

Host sends: [02] [00] [01] [E6] [17]

Coin s. respond: [01] [02] [02] [00] [00] [FB]

Example of message string for coin selector(*address 2*) with programmed positions from 1 to 6, after receiving command to enable acceptance of all 16 position is:

Host sends: [02] [00] [01] [E6] [17]

Coin s. respond: [01] [02] [02] [00] [3F] [00] [BC]

First byte represent the mask for coin positions 1 to 8 and second for 9 to 16.

Coin channels(*positions*) that are not programmed are always represented as zero bit!

#### 6.5.2.12 Command 229 [hexE5], Read buffered credit or error codes

This is the most important command used by host to detect import of coins in to a machine and to report eventual errors. As previously mentioned coin selectors store all events in volatile memory called credit buffer. Buffer has 5 level and use two bytes for each event. In first byte coin position or coin value<sup>18</sup> is stored. The second byte point to a sorter path or indicate error code.

If during coin acceptance any error occurs, stored value of coin position is 0, hex [00].

Error codes supported in ALBERICI coin selectors AL55/66 are shown in table 6.4.

<sup>16</sup> Positions are sent by coin selector during reading credit buffer or error codes (*header 229*)

<sup>17</sup> Coin selector for Italian gambling machines

<sup>18</sup> If coin selector use CVF (*Coin Value Format*)

Code d.	Code h.	Error	Coin rejected
0	00	Null event	No
1	01	Reject coin (not recognized)	Yes
2	02	Inhibited coin (master inhibit)	Yes
3	03	Multiple window (fraud or similar coin)	Yes
5	05	Validation (measuring) time out	Yes
6	06	Credit sensor (recognition to opto 2) time out	Possible
8	8	Second close coin	Yes/both
16	10	Credit sequence error (Yo-yo)	No
18	12	Coin to fast (opto 2 minimum time not elapsed)	No
19	13	Coin to slow (opto 2 time out)	No
128	80	Inhibited coin (position 1)	Yes
...	...	Inhibited coin (position n)	Yes
143	8F	Inhibited coin (position 16)	Yes
255	FF	Unspecified alarm code	-

Table 6.4 Acceptance error codes

Coin selectors also use one eight bit counter<sup>19</sup> that is incremented each time a new coin is detected. At the same time data in coin credit buffer are shifted two position to the right. When counter reaches the value of 255 it toggle to a value 1 and continue to increment on each event. Event counter is set to value "0" after each power-up or acceptance of reset command. The first two byte (LSB) in coin credit buffer always contain the data of last event. Host software must read event counter and coin credit buffer data in period short enough to prevent the loss of coin data<sup>20</sup>. Message format is:

Host sends: [Dir] [00] [00] [E5] [Chk]

Coin s. respond: [01][0B] [Dir] [00] [Ev.cnt.][coin code 1][dir/err] [coin code 2][dir/err] . . .  
. . . [coin code 5][dir/err] [Chk]

Examples of message string for coin selector(address 2) after coin insertions:

Host sends: [02] [00] [00] [E5] [18] Polling minimum each 500 ms

Coin s. respond: [01] [0B] [02] [00] [00][00][00][00][00][00][00][00][00][00] [F2]

The respond after power-up or reset

Coin s. respond: [01] [0B] [02] [00] [01][01][02][00][00][00][00][00][00][00] [EE]

First event, coin position 1, sorter path 2 accepted

Coin s. respond: [01] [0B] [02] [00] [02][02][01][ 01][02][00][00][00][00][00] [EA]

Second event, coin position 2, sorter path 1 accepted

Coin s. respond: [01] [0B] [02] [00] [03][00][02][02][01][01][02][00][00][00] [E7]

Third event, coin rejected due to master inhibit active

Coin s. respond: [01] [0B] [02] [00] [04][00][83][ 00][02][02][01][01][02][00][00] [63]

Forth event, coin position 4 inhibited and rejected

From example we can notice shifting of data in the coin credit and error buffer and increment of event counter.

### 6.5.2.13 Command 228 [hexE4], Modify master inhibit status

This command is used to inhibit acceptance of all coins and has same effect as command modify inhibit status with sent with two bytes of zeros. Host sends only one byte of data. If first bit (LSb) is set to "0" coin selector is inhibited. Bits 1 to 7 has no influence to coin selector. Message format is:

Host sends: [Dir] [01] [01] [E4] [Mask.] [Chk]

Coin s. respond: [01] [00] [Dir] [00] [Chk] ACK

**Initially coin selectors are programmed with acceptance enabled.**

**Change is stored in RAM location .**

**On customer demand it is possible to set inhibition as default .**

Example of message string to inhibit the acceptance for coin selector(address 2) is:

Host sends: [02] [01] [01] [E4] [00] [18]

Coin s. respond: [01] [00] [02] [00] [FD] ACK

After that coin selector acceptance will be inhibited till reset or next instruction that will change master inhibit status.

<sup>19</sup> Event counter

<sup>20</sup> See command 249 Request polling priority



#### 6.5.2.14 Command 227 [hexE3], Request master inhibit status

Coin selector respond with one byte data information of main inhibit status.

Only first (*LSb*) bit is used. If bit 0 is "1" acceptance is enabled, and if bit 0 is "0" coin selector is inhibited and acceptance is disabled.

Other bits has no meaning and always read as "0". Message format is:

Host sends: [Dir] [00] [00] [E3] [Chk]

Coin s. respond: [01] [01] [Dir] [00] [Mask.] [Chk]

Example of message string for coin selector(*address 2*) after power-up is:

Host sends: [02] [00] [01] [E3] [1A]

Coin s. respond: [01] [01] [02] [00] [01] [FB] Acceptance enabled (*default*)

Example of message string for coin selector(*address 2*) after activation of master inhibit<sup>21</sup> is:

Host sends: [02] [00] [01] [E3] [1A]

Coin s. respond: [01] [01] [02] [00] [00] [FC] Coin selector inhibited

#### 6.5.2.15 Command 226 [hexE2], Request insertion counter

Coin selector respond with three bytes of insertion counter data.

First byte is LS byte of three byte counter in RAM. Insertion counter is set to zero after power up or reset command. It is incremented each time a new coin is inserted in to coin acceptor. Message format is:

Host sends: [Dir] [00] [00] [E2] [Chk]

Coin s. respond: [01] [03] [Dir] [00] [Cunt1-LSB] [Cunt2] [Cunt3-MSB] [Chk]

Example of message string for coin selector(*address 2*) after power-up is:

Host sends: [02] [00] [01] [E2] [1B]

Coin s. respond: [01] [03] [02] [00] [00] [00] [FA]

#### 6.5.2.16 Command 225 [hexE1], Request accept counter

Coin selector respond with three bytes of acceptance counter data.

First byte is LS byte of three byte counter in RAM. Acceptance counter is set to zero after power up or reset command. It is incremented each time a new coin pass acceptance sensor<sup>22</sup>. Message format is:

Host sends: [Dir] [00] [00] [E1] [Chk]

Coin s. respond: [01] [03] [Dir] [00] [Cunt1-LSB] [Cunt2] [Cunt3-MSB] [Chk]

Example of message string for coin selector(*address 2*) after power-up is:

Host sends: [02] [00] [01] [E1] [1C]

Coin s. respond: [01] [03] [02] [00] [00] [00] [FA]

.....

#### 6.5.2.22 Command 210 [hexD2], Modify sorter paths

With this command host is able to change direction of coins in sorter if sorter is supported. Host sends two bytes of data to select the coin position and sorter path (*direction of exit*). First byte of data (*LSB*) represent coin position and second byte of data point to sorter path. ALBERICI coin selectors has support for most existing sorters that has direct drive of coils from coin selector with open collector transistor. Most common are 3 or 4 way sorter with two coils<sup>23</sup>, but recently 5 way sorters<sup>24</sup> with 3 coils are in use. Message format is:

Host sends: [Dir] [02] [01] [D2] [Coin pos.] [Sort.Path] [Chk]

Coin s. respond: [01] [00] [Dir] [00] [Chk] ACK if sorter path is possible to set

Coin s. respond: [01] [00] [Dir] [05] [Chk] NAK if coin selector does not support setting

**Initially all coin position has sorter paths set to direction 1 hex[01].**

**If sorter is not supported, sorter path is set initially to 0 hex[00]!**

If host sends command to modify sorter path that is not existent or for coin not programmed, the coin selector will respond with message NAK. Ex. of message string for coin selector(*address 2*) redirection of coin **pos. 1** in to **path 2** is:

Host sends: [02] [02] [01] [D2] [01] [02] [26]

Coin s. respond: [01] [00] [02] [00] [FD] ACK

<sup>21</sup> Using command 210, Modify master inhibit status

<sup>22</sup> Credit sensor

<sup>23</sup> Maximum current consumption for each coil is 500 mA

<sup>24</sup> 5-way VARIANT sorter from ALBERICI

After acceptance of command, accepted coins with position 1 will exit in direction 2 of the sorter. The path or direction 1 is usually one without activation of any coil.

Different coil activation schematics is possible to program by setting the sorter type.

#### 6.5.2.23 Command 209 [hexD1], Request sorter paths

Host send one byte of coin position and coin selector respond with one byte of sorter path. Message format is:

Host sends: [Dir] [01] [00] [D1] [Coin pos.] [Chk]

Coin s. respond: [01] [01] [Dir] [00] [Sort.Path] [Chk]

Example of message string for coin selector(*address 2*) for initial sorter path 1 of coin position 1:

Host sends: [02] [01] [01] [D1] [01] [2A]

Coin s. respond: [01] [01] [02] [00] [01] [FB]

Example of message string for coin selector (*address 2*) for sorter path 2 of coin position 1:

Host sends: [02] [01] [01] [D1] [01] [2A]

Coin s. respond: [01] [01] [02] [00] [02] [FA]

**If host request sorter path for non programmed coins or non existent position<sup>25</sup>, the coin selector will respond with message NAK !**

.....

#### 6.5.2.26 Command 196 [hexC4], Request creation date

Coin selector respond with two byte of data that represent codified date of production. Date of production is codified in so called *RTBY (Relative To Base Year)*<sup>26</sup> format. Message format is:

Host sends: [Dir] [00] [01] [C4] [Chk]

Coin s. respond: [01] [02] [Dir] [00] [LSB] [MSB] [Chk]

Example of message string for coin selector (*address 2*) with date of production 05.07.2003 is:

Host sends: [02] [00] [01] [C4] [39]

Coin s. respond: [01] [02] [02] [00] [E5] [06] [10]

ALBERICI coin selectors has date of production written in monitor part of MCU FLASH memory which is not possible to change without factory FLASH reprogramming.

#### 6.5.2.27 Command 195 [hexC3], Request last modification date

Coin selector respond with two byte of data that represent codified date of last modification of software<sup>27</sup>. Date of modification is codified also in RTBY format.

Message format is:

Host sends: [Dir] [00] [01] [C3] [Chk]

Coin s. respond: [01] [02] [Dir] [00] [LSB] [MSB] [Chk]

Example of message string for coin selector (*address 2*) with date of modification 23.07.2003 is:

Host sends: [02] [00] [01] [C3] [3A]

Coin s. respond: [01] [02] [02] [00] [F7] [06] [FE]

NOTICE: after each up-grade of coin selector program FLASH memory date will correspond to software modification date, not to the actual date of up-grade!

#### 6.5.2.28 Command 194 [hexC2], Request reject counter

Coin selector respond with three bytes of reject counter data.

First byte is LS byte of three byte counter in RAM. Reject counter is set to zero after power up or reset command. It is incremented each time a coin is inserted but not recognized. Message format is:

Host sends: [Dir] [00] [00] [C2] [Chk]

Coin s. respond: [01] [03] [Dir] [00] [Cunt1-LSB] [Cunt2] [Cunt3-MSB] [Chk]

Example of message string for coin selector(*address 2*) after power-up is:

Host sends: [02] [00] [01] [C2] [3B]

Coin s. respond: [01] [03] [02] [00] [00] [00] [00] [FA]

---

<sup>25</sup> Position bigger than 16

<sup>26</sup> For details see cctalk protocol, document cctalk44-2.pdf

<sup>27</sup> Up-grade of FLASH program memory

### 6.5.2.29 Command 193 [hexC1], Request fraud counter

Coin selector respond with three bytes of fraud coins counter data.

First byte is LS byte of three byte counter in RAM. Fraud counter is set to zero after power up or reset command. It is incremented each time a coin acceptor recognize coin that is programmed as "fraud" coin<sup>28</sup>. Message format is:

Host sends: [Dir] [00] [00] [C1] [Chk]

Coin s. respond: [01] [03] [Dir] [00] [Cunt1-LSB] [Cunt2] [Cunt3-MSB] [Chk]

Example of message string for coin selector(address 2) after power-up is:

Host sends: [02] [00] [01] [C1] [3C]

Coin s. respond: [01] [03] [02] [00] [00] [00] [00] [FA]

---

### 6.5.2.30 Command 188 [hexBC], Request default sorter path

For ALBERICI coin selectors AL55/66 the default sorter path is always hex[01].

Example of message string for coin selector(address 2) is:

Host sends: [02] [00] [01] [BC] [41]

Coin s. respond: [01] [01] [02] [00] [01] [FB]

### 6.5.2.32 Command 184 [hexB8], Request coin ID

Host use this command at initialization process to build table for each coin position value. If coin selector uses CVF it is obsolete command.

Host send one byte data of coin position and coin selector respond with 6 byte ASCII string of characters that describes the requested coin position.

Message format is:

Host sends: [Dir] [01] [01] [B8] [Coin pos] [Chk]

Coin s. respond: [01] [06] [Dir] [00] [a1][a2][a3][a4][a5][a6] [Chk]

Example of message string for coin selector(address 2) and coin position 1(2 Euro) is:

Host sends: [02] [01] [01] [B8] [01] [43]

Coin s. respond: [01] [06] [02] [00] [45][55][32][30][30][41] [8A] Coin 'EU200A'

For none-programmed position the ASCII string is: '.....'.

Example of message string for coin selector(address 2) and coin position 12 that is not programmed is:

Host sends: [02] [01] [01] [B8] [0C] [38]

Coin s. respond: [01] [06] [02] [00] [2E][ 2E][ 2E][ 2E][ 2E][ 2E] [E3] Coin not programmed

### 6.5.2.35 Command 170 [hexAA], Request base year

Coin selector respond with four byte ASCII string of character representing the base year for calculation of exact date of production. Message format is:

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

Coin s. respond: [01] [04] [Dir] [00] [a1][a2][a3][a4] [Chk]

For ALBERICI coin selectors base year is 2000.

Example of message string for coin selector(address 2) is:

Host sends: [02] [00] [01] [AA] [53]

Coin s. respond: [01] [04] [02] [00] [32][30][30][30] [37]

## 6.5.3 MDCES command headers

MDCES stands for **M**ulti-**D**rop **C**ommand **E**xtension **S**et, or so called Multi-drop buss commands. Multi-drop buss commands gives additional functionality to systems that require change of address for devices in cctalk network.

Some of commands has different message format than usual cctalk message.

Commands are:

- Address poll
- Address clash
- Address change
- Address random

Because host always use address 1 and address 0 is for broadcast message all commands that changes the address should not accept this settings.

**All changes are stored in non-volatile memory, EEPROM !**

---

<sup>28</sup> Coins with close recognition parameters sometime called "killer coin or channel"

### 6.5.3.1 Command 253 [hexFD], Address poll

This is a broadcast message used by host to determinate all address of device attached on cctalk network. Coin selector respond with only one byte (*non-standard message format*), after a delay that is proportional to address value multiplied with 4 milliseconds. Message format is:

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

Coin s. respond: **Dly -> [Address]**

Example of message string for coin selector(*address 2*) is:

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

Coin s. respond: **Dly=8 ms -> [02]** Address is 2

Example of message string for coin selector (*address 250*) is:

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

Coin s. respond: **Dly=1 s -> [FA]** Address is 250

### 6.5.3.2 Command 252 [hexFC], Address clash

Command Address clash has same respond from coin selector but host issue this command with specific device address. Coin selector respond with only one byte (*non-standard message format*), after a random value of time delay to prevent collision if two devices share same address. Message format is:

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

Coin s. respond: **Random Dly -> [Address]**

Example of message string for coin selector(*address 2*) **AL06V-c** is:

Host sends: **[02] [00] [01] [FC] [01]**

Coin s. respond: **Random Dly -> [02]** Address is 2

