AL66-AL55 Selettori elettronici di monete

Manuale d'uso

Rev. 1.02

Manuale d'uso

CE

Alberici®
CASH SOLUTIONS

Progettazione e produzione di sistemi di pagamento e accessori per macchine Gaming, Vending e Car-Wash

Via Cà Bianca, 421 - 40024
Castel San Pietro Terme (BO) - Italy

Tel.: +39.051.944300
Fax.: +39.051.944594

Web: www.alberici.net
E.mail: info@alberici.net

Sommario

1. Descrizione generale

Il selettore elettronico di monete AL66 appartiene alla terza generazione di selettori elettronici Alberici.
E' compatibile con la precedente generazione di selettori AL05 e AL06, sia elettricamente che meccanicamente. E' ugualmente compatibile con altri selettori da 3 ½" aventi connettore IDC 10 poli. E' possibile cambiare la polarità dell'alimentazione in ingresso per renderlo sostituibile a selettori di altro tipo (vedere ad es. EMULATORE A mod. 4, ed EMULATORE M mod. 3)

L'Utilizzatore può cambiare le funzioni operative del selettore Alberici per adattarlo a varie e diverse applicazioni, utilizzando il nostro software di programmazione gratuito. La modifica si può eseguire impostando una delle funzioni standard descritte in questo documento, oppure creando la propria funzione personalizzata attraverso l'interfaccia seriale e la finestra del terminale. Il Cliente può anche ordinare una funzione speciale creata da esperti di fabbrica.
I selettori di monete possono essere forniti pre-programmati, oppure possono essere ri-programmati dal Cliente mediante clonazione.

La terza generazione di selettori elettronici possiede un sistema di misurazione avanzato, che comprende tre coppie di sensori magnetici, più un rilevatore ottico del diametro della moneta.
Il nucleo del selettore è il microcontrollore "Freescale" a 8 bit di nuova generazione, con memoria FLASH da 36 kB ed elevata immunità ai disturbi elettromagnetici (e per questo è diffuso nel settore "automotive").
Il numero dei sensori di controllo del percorso della moneta dipende dalla versione[1].

Viene supportata la modalità di risparmio energetico, e c'è la possibilità di gestire la modalità di riattivazione da "sleep-state".

In certe versioni speciali (ad es. per cabina telefonica) è possibile spegnere l'alimentazione applicando il segnale di trigger a uno dei piedini del connettore a 10 poli. L'assorbimento di corrente è ulteriormente ridotto in stand-by e in modalità di risparmio energetico.
Le tolleranze rispetto alla tensione di alimentazione sono ampliate rispetto alla generazione precedente, ed è migliorata la resistenza alle IEM (interferenze elettro-magnetiche).

E' supportato l'aggiornamento del firmware, e il tempo necessario per l'aggiornamento completo è ridotto a ca. 10 secondi.

La nuova generazione di selettori elettronici offre l'opzione di trasferimento protetto dei dati e l'autenticazione sicura mediante crittografia RSA (*algoritmo asimmetrico con chiave pubblica e privata*).

N.B. le gettoniere AL66 ccTalk sono conformi ai requisiti di non-modificabilità prescritti dalla Legge Italiana per le macchine da gioco (Legge n. 289).

[1] Il numero delle coppie di sensori è 1, 2 o 3 nella versione più avanzata e sicura.

2. Configurazioni meccaniche

I selettori di monete AL55 e AL66 sono prodotti in quattro modelli:

- Modello V
- Modello I (V invertito)
- Modello K (frontalino standard)
- Modello S (frontalino mini)

2.1 Modello V

Questo modello è meccanicamente compatibile con le gettoniere meccaniche da 3 ½”.
Il selettore si integra in qualunque supporto standard a incastro o a scatto.
L’introduzione della moneta è dall’alto, le uscite per le monete accettate e per quelle scartate sono sul lato inferiore (cfr. figura 2.1). Il percorso della moneta accettata è più lungo del percorso della moneta scartata, quindi la velocità di accettazione è di max. 3 monete al secondo.
La corsia della moneta accettata esegue una doppia curva e ciò rende difficile l’eventuale tentativo di ripescaggio (moneta con filo). Inoltre il modello V è provvisto di due meccanismi trancia-filo e di un taglia-filo.

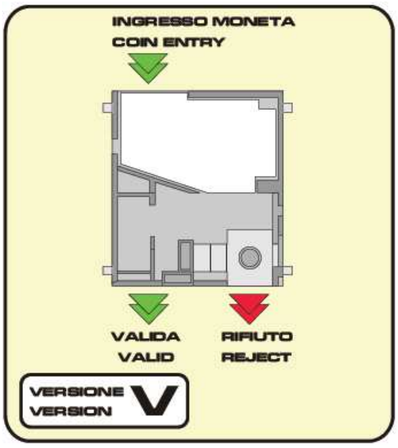


Fig. 2.1 Percorso monete versione V

2.2 Modello inverso I

Questo modello è simile al modello V, salvo che l’uscita delle monete accettate e quella delle monete scartate sono invertite (cfr. figura 2.2)
Il percorso della moneta accettata è breve, quindi la velocità di accettazione può raggiungere le 4 monete al secondo.
Questa versione è spesso usata nei parchimetri.
Le monete accettate escono verticalmente così che sono installati solo un meccanismo taglia-filo e un trancia-filo.

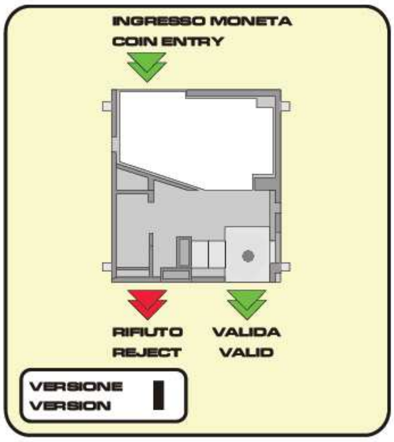


Fig. 2.2 Percorso monete versione I

2.3 Modello K (per frontalino standard K)

Questo modello è usato nelle giostre per bambini.
Il selettore è montato su una piastra che funge anche da supporto. Ciò comporta una maggiore esposizione del selettore alle condizioni esterne e ai tentativi di frode.
L'introduzione della moneta avviene dall'alto, l'uscita della moneta accettata avviene verticalmente sul lato inferiore, mentre la moneta scartata esce frontalmente dalla parte bassa del frontalino (*cfr. figura 2.3*).
Il percorso di accettazione è corto e quindi la velocità di accettazione può raggiungere le 4 monete al secondo.
Le monete accettate escono verticalmente così che sono sufficienti solo un meccanismo taglia-filo e uno trancia-filo.

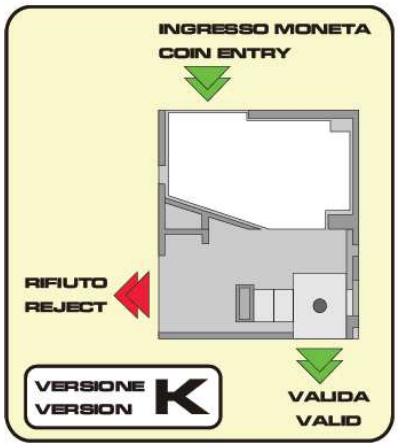


Fig. 2.3 Percorso monete versione K

2.4 Modello S (per frontalino mini S o C)

E' molto simile al modello K, ma la piastra frontale è più corta e occupa uno spazio minore. La fessura d'introduzione si estende anche frontalmente (*cfr. Figura 2.4*).
Di conseguenza la vulnerabilità di questo selettore alle condizioni esterne e ai tentativi di frode è maggiore che sugli altri modelli.

Fig. 2.4 Percorso monete versione S

3. Connessioni

Il selettore si collega alle periferiche e alla scheda macchina attraverso i connettori illustrati in Figura 3.1.

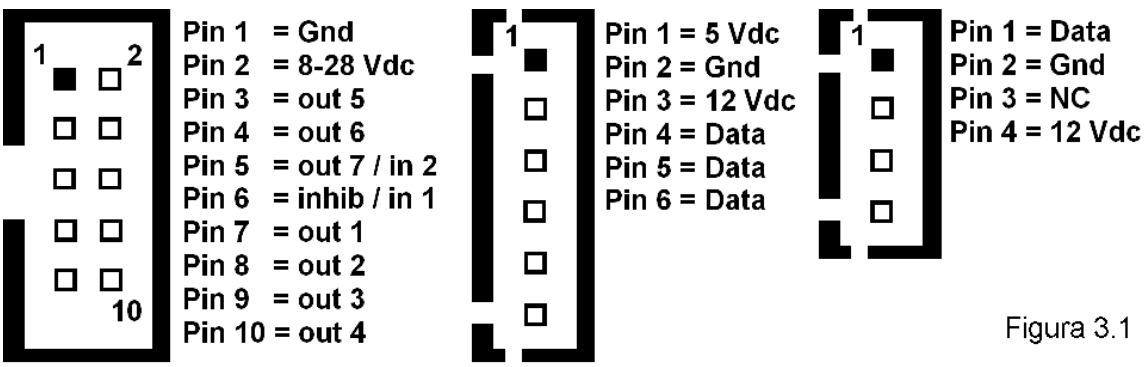
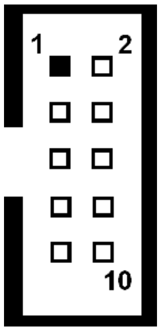


Figura 3.1

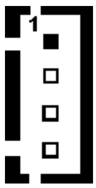
Il connettore interfaccia **X1** è una spina IDC 10 poli, il cui schema è visibile in Figura 3.2. E' composto da: 2 contatti (*pin 1 positivo e pin 2 comune o massa*) per l'alimentazione, 6 uscite "open collector" con stato attivo LOW (*pin 3,4,7,8,9 e 10*), da un contatto di entrata con stato attivo HIGH *(pin 6, generalmente usato per inibire l'accettazione - modificabile in stato attivo LOW)* e da un piedino a doppia funzione (*pin 5*), che può essere usato sia come uscita open collector sia come entrata supplementare.

A richiesta, è possibile ricevere il selettore con la polarità dell'alimentazione invertita, per emulare quella dei selettori di tipo spagnolo.



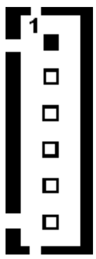
nr.	Descrizione
1	Gnd
2	8-26 Vdc
3	Out 5 / bobina separatore B
4	Out 6 / bobina separatore A
5	Out 7 (totalizzatore) / In 2
6	In 1 (inibizione)
7	Out 1
8	Out 2
9	Out 3
10	Out 4 / bobina separatore C

Il connettore **X3** a 4 poli è usato per la comunicazione seriale **cctalk®**[2] con la scheda macchina. Il protocollo è predisposto per funzionamento in modalità "slave", ed è descritto nel capitolo 6 del presente documento.



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

Il connettore **X2** a 6 poli (solo AL66) è per il collegamento al display o per il modulo di crittografia[3]. Diversi display con protocollo di comunicazione SPI o I²C bus sono supportati.



nr.	Descrizione
1	5 Vdc
2	Gnd
3	12 Vdc
4	Dati
5	Dati
6	Dati

Il display va impostato in fabbrica e/o modificato tramite il software Alberici. I seguenti modelli sono supportati:

- MC 14499 4 digit compatibile con RM924S SECI o G-51.1092 NRI
- MC 14489 5 digit compatibile con G-53.0747 NRI
- MAX 7219 6 digit AL066 ALBERICI
- M643 8 digit LCD

[2] I dettagli del protocollo (Money Controls) sono disponibili sul sito: http://www.cctalk.org

[3] MCU addizionale SPI crypto

4. Versioni funzionali

Esistono diversi tipi di funzioni e/o versioni che il gestore può autonomamente impostare attraverso un programma di supporto (*programmatore*) e un PC.
Le versioni disponibili sono:
1. Validatori
2. Totalizzatori
3. Temporizzatori
4. Multiprezzo
5. Protocollo cctalk internazionale modificabile
6. Protocollo cctalk immodificabile (Italy & Germany)

Il protocollo di comunicazione cctalk e' riportato in inglese nella seconda parte del presente manuale.

4.1 Validatori

A questo gruppo appartengono le gettoniere con le sei uscite poste in parallelo.
A ciascuna uscita può essere associato uno (o più) tra i dodici canali disponibili, e può lavorare nella modalità *impulso singolo o multi impulso.*
Sono disponibili anche nella versione autoprogrammabile con la possibilità di programmazione di 6 canali tramite DIP-switch(*vedi descrizione nel capitolo 5 OPZIONI*).
L'impostazione del tempo di durata del segnale attivo come la programmazione dei canali viene effettuata tramite il programmatore.
La disabilitazione della singola uscita o canale si effettua tramite il DIP-switch direttamente sulla gettoniera o tramite il programmatore.
Portando una tensione da 3 VDC a 30 VDC sul pin 6 (oppure *5, se e' stato scelto*) viene bloccata (inibita) l'accettazione di tutte le monete (***Inhibit***).

Esistono differenti versioni:

- Standard parallelo. E' il più usato, emette un impulso dall'uscita relativa alla moneta transitata, per ogni moneta accettata.
- Parallelo combinatorio (EVA standard). Per dettagli cfr. documento EVA Spec. 01.pdf. L'uscita 1 è sempre attiva per indicare che si tratta di una comunicazione combinatoria. Le uscite da 2 a 7 in combinazione binaria (0-1) indicano fino a 15 diversi conii.
- Multi-impulso parallelo. Diverse monete di valore multiplo sono programmate sulla stessa uscita. Questa emette un impulso se la moneta accettata è quella base, o due se la moneta accettata ha valore doppio, e così via.
- Parallelo con protocollo spagnolo. Utilizzato a volte in macchine di fabbricazione spagnola, con comando particolare dell'inibizione e del separatore.

4.2 Totalizzatori

I totalizzatori usano una sola uscita multi-impulso (pin 5), ed emettono un impulso ogni volta che l'importo accettato raggiunge il valore del credito.
Tale valore può essere programmato mediante il software di programmazione o settato tramite i dip-switch a bordo. E' possibile definire anche la durata dell'impulso (valore tipico 100 ms ON, e 200 ms OFF).
E' inoltre possibile programmare i livelli e i valori di bonus.
Una delle uscite può essere usata per comandare un contatore esterno.
Se è selezionata l'opzione display (solo AL66), è possibile visualizzare l'importo in corso di accumulazione.
L'uscita si trova sul pin 5; l'inibizione è sul pin 6. E' possibile spostare l'uscita sul pin 9.
Il totalizzatore può essere configurato per attivare l'impulso solo su richiesta: è necessario in tal caso programmare prevedere la richiesta sul pin 6, sotto forma di ritorno del segnale di inibizione al livello iniziale.

4.3 Timer

Sono le gettoniere che forniscono in uscita un impulso temporizzato la cui durata è proporzionale al valore inserito.
Esistono due versioni di base per i temporizzatori:
• **Timer progressivo:** quando il valore accumulato raggiunge il valore di un credito viene attivata l'uscita per un determinato periodo di tempo. Aggiungendo monete, proporzionalmente viene incrementato il tempo di attivazione. Il tempo viene calcolato moltiplicando il valore accumulato per il moltiplicatore programmato.
• **Timer a richiesta**: l'attivazione dell'uscita avviene portando una tensione alta sul pin d'entrata (pin 6), sempre se il valore accumulato raggiunge il valore di un credito.
Il periodo di durata dell'impulso e' sempre lo stesso, e, se al termine esistono ancora crediti non realizzati, portando di nuovo una tensione all'entrata viene attivato l'impulso seguente. Nei timer a richiesta, se viene introdotto denaro dopo l'avvio dell'erogazione, non si avrà addizione progressiva all'erogazione in corso. Se l'importo è sufficiente, sarà possibile avviare una nuova erogazione al termine di quella in corso.
L'uscita si trova sul pin 5; l'inibizione è sul pin 6. E' possibile spostare l'uscita sul pin 9.
Una delle uscite può essere usata per comandare un contatore esterno.
Se è selezionata l'opzione display (solo AL66), è possibile visualizzare l'importo in corso di accumulazione, o il tempo, o il resto.

4.4 Multi-prezzo

Le uscite restano attivate fino a quando il valore accumulato non arriva al valore impostato *(prezzo).* Ogni uscita può essere impostata con un prezzo differente.
Raggiungendo il prezzo determinato viene attivata l'uscita corrispondente.
La gettoniera multi-prezzo può essere impostata solamente in fabbrica.
Esistono due modi di gestione delle uscite:
- Il primo modo e' che la linea del minor prezzo si disattivi quando il valore impostato raggiunge il valore dell seguente prezzo (piu' alto) programmato.
- Il secondo modo e' che rimangono attive tutte le linee con il prezzo raggiunto.

Dopo che la vendita è stata effettuata, sottraggono sempre il valore della massima linea attiva di vendita.
Quando e' stato raggiunto il prezzo massimo, la gettoniera non accetta più le monete!
Dopo l'effettuata vendita, l'host manda all'entrata un impulso di **reset** con il quale annulla il valore del prezzo dell'uscita attiva (e lo detrae da quello accumulato). In tal caso, se il valore accumulato rimanente non è sufficiente per una erogazione, l'uscita viene disattivata.
E' possibile trattenere o cancellare il resto (subito o dopo un certo periodo), se e' minore del prezzo minimo (regolabile).

4.4.1 Multi-prezzo (6 linee) Questo tipo di selettore ha 6 linee di uscita programmabili con prezzi diversi. L'inibizione è sul pin 6 (input 1); la linea di reset è sul pin 5 (input 2). L'impulso di reset deve essere attivo per almeno 50 ms per ottenere il reset della linea. E' possible collegare un display (solo AL66) sul connettore 6 poli.

4.4.2 Dual-price Questo tipo di selettore usa solo 2 linee di prezzo. Inibizione e reset sono entrambi sul pin 6 (input 1). L'inibizione è attiva alta durante l'erogazione e va bassa alla fine; in quel momento la linea di prezzo viene resettata. E' possible collegare un display (solo AL66) sul connettore 6 poli. Il banco di Dip-switch SW1 è usato per programmare i prezzi.

4.4.3 Foto-copiatrice
Questa versione usa solo una uscita per effettuare la vendita, e si cancella (azzera) quando la gettoniera riceve un determinato numero d'impulsi (regolabili). Inibizione e reset sono entrambi sul pin 6 (input 1), l'uscita è sul pin 9 (output 3); l'uscita per il valore di resto è sul pin 10 e quella per il contatore è sul pin 8.

5. Funzioni speciali

Sono le seguenti:
- Modalità risparmio energetico
- Modalità di auto-programmazione
- Separazione monete
- Display (solo AL66)
- Gestione del resto
- Inibizione frode

5.1 Modalità risparmio energetico

In questa modalità l'assorbimento è ridotto a pochi million-ampere.
Questa modalità deve essere richiesta espressamente in anticipo e abilitata in fabbrica.
L'utente può comunque disabilitare la funzione di risparmio energetico e programmare il periodo di attesa prima che il selettore vada in sleep-mode (tempo min. 10 sec, tempo max. 254 sec).
Per ri-programmare il periodo di stand-by, l'utente deve resettare il selettore e modificare il tempo prima che il selettore torni in sleep-mode.

Sono disponibili due modalità distinte:
- Standard
- Con auto-risveglio

5.1.1 Standard

Il selettore va risvegliato applicando 5-30 V al pin 6 (input 1) per almeno 1 msec.. In ca. 50 msec. Il selettore è pronto a funzionare.
Ad ogni moneta inserita, il conteggio per il rientro in modalità di risparmio energetico viene azzerato.

5.1.1 Con auto-risveglio

Il risveglio avviene all'inserimento di una moneta: la prima moneta inserita, che risveglia il selettore, viene restituita.

5.2 Modalità di auto-programmazione

Questo tipo di gettoniera si può programmare senza l'ausilio del PC.
L'auto-programmazione viene realizzata tramite l'uso del banco di **Dip Switch** posto sul lato connettori. In alternativa, si può usare il software Alberici AL66 con il kit GETTO-600, per programmare il selettore via computer.
Al canale d'uscita programmato (switch in ON) arriva un impulso ad ogni passaggio della moneta programmata per essere riconosciuta nel suddetto canale.

ISTRUZIONI DI AUTOPROGRAMMAZIONE AL66 STD

A) ***PER ESEGUIRE LA PROGRAMMAZIONE***:

1. A macchina spenta porre gli Switch dei banchi SW1 e SW2 in posizione OFF

2. accendere l'alimentazione: la spia verde sul lato posteriore della gettoniera lampeggia

3. Banco-switch SW1- scegliere il n° di canale per il taglio di moneta da programmare, e selezionare gli switch corrispondenti (vedi tabella: 'Dip-switch Banco SW1'). Ad es., il canale 4 richiede DS 1 e DS2 in posizione ON

N.B.: *Se la moneta è da accettare, DS 5 va su OFF e DS6 su ON. Se invece la moneta è da rifiutare come falsa, DS5 va su ON e DS6 su OFF.*

Dip-switch Banco SW1						
NUMERO CANALE	*DS1*	*DS2*	*DS3*	*DS4*	*DS5*	*DS6*
Canale 1	OFF	OFF	OFF	OFF	OFF	ON
Canale 2	ON	OFF	OFF	OFF	OFF	ON
Canale 3	OFF	ON	OFF	OFF	OFF	ON
Canale 4	ON	ON	OFF	OFF	OFF	ON
Canale 5	OFF	OFF	ON	OFF	OFF	ON
Canale 6	ON	OFF	ON	OFF	OFF	ON
Canale 7	OFF	ON	ON	OFF	OFF	ON
Canale 8	ON	ON	ON	OFF	OFF	ON
Canale 9	OFF	OFF	OFF	ON	OFF	ON
Canale 10	ON	OFF	OFF	ON	OFF	ON
Canale 11	OFF	ON	OFF	ON	OFF	ON
Canale 12	ON	ON	OFF	ON	OFF	ON
Canale 13	OFF	OFF	ON	ON	OFF	ON
Canale 14	ON	OFF	ON	ON	OFF	ON
Canale 15	OFF	ON	ON	ON	OFF	ON
Canale 16	ON	ON	ON	ON	OFF	ON
Marca canale moneta falsa	….	….	….	….	ON	OFF
Cancella canale	….	….	….	….	OFF	OFF
Reset impostazione fabbrica	ON	ON	ON	ON	ON	ON

4. Banco-switch SW2- scegliere il valore da attribuire al taglio di moneta che si vuole programmare nel canale, e selezionare gli switch corrispondenti (vedi tabella in pag. seguente: 'Dip-switch Banco SW2'). Es.: il valore 40 richiede DS 1 e DS4 in pos. ON:

Dip-switch Banco SW2						
VALORE DA ASSEGNARE AL CANALE	*DS1*	*DS2*	*DS3*	*DS4*	*DS5*	*DS6*
Uguale all'esistente	OFF	OFF	OFF	OFF	X	X
1	ON	OFF	OFF	OFF	X	X
2	OFF	ON	OFF	OFF	X	X
4	ON	ON	OFF	OFF	X	X
5	OFF	OFF	ON	OFF	X	X
8	ON	OFF	ON	OFF	X	X
10	OFF	ON	ON	OFF	X	X
20	ON	ON	ON	OFF	X	X
25	OFF	OFF	OFF	ON	X	X
40	ON	OFF	OFF	ON	X	X
50	OFF	ON	OFF	ON	X	X
80	ON	ON	OFF	ON	X	X
100	OFF	OFF	ON	ON	X	X
125	ON	OFF	ON	ON	X	X
200	OFF	ON	ON	ON	X	X
250	ON	ON	ON	ON	X	X

5. Introdurre una dopo l'altra 15 monete; la bobina della gettoniera si attiva per due volte consecutive (doppio scatto), a conferma dell'accettazione della programmazione

6. Spegnere l'alimentazione. Porre in posizione ON tutti i dip-switch del banco SW1 e in posizione OFF tutti i dip-switch del banco SW2.

7. Per programmare altre monete in altri canali, porre tutti gli Switch dei banchi SW1 e SW 2 in OFF, e ripetere le operazioni dal punto 2 fino al punto 6

N.B.: una volta programmato un canale come "falso", non è più possibile ri-attivarlo o ri-programmarlo manualmente: è necessario utilizzare il software di programmazione. E' però possibile riportare la gettoniera alle impostazioni di fabbrica (vedi oltre).

B) ***PER COLLAUDARE LA PROGRAMMAZIONE ESEGUITA***:

1. Porre in ON tutti i dip-switch del banco SW1 e quelli del banco SW2.
2. Accendere l'alimentazione e verificare l'accettazione o il rifiuto delle monete programmate.

C) ***PER CANCELLARE UN CANALE PROGRAMMATO***:

1. A macchina spenta porre gli Switch del banco SW1 e quelli del banco SW2 in posizione OFF
2. accendere l'alimentazione
3. **Banco-switch SW1-** scegliere il n° di canale per il taglio di moneta da programmare, e selezionare gli switch corrispondenti (vedi tabella: 'Dip-switch Banco 1'). Ad es., il canale 7 richiede DS 2 e DS3 in posizione ON. *Posizionare entrambi il DS 5 e il DS 6 su OFF.*
4. Introdurre una dopo l'altra 3 monete di qualunque valore; dopo la prima, la spia rossa lampeggia, dopo la terza la bobina della gettoniera si attiva per due volte consecutive (doppio scatto), e la spia verde lampeggia a conferma dell'accettazione del ripristino.
5. Spegnere l'alimentazione.

N.B.: *una volta cancellato un canale, non è più possibile ri-attivarlo o ri-programmarlo manualmente: è necessario utilizzare il software di programmazione AL66 WinProg Alberici. E' però possibile riportare la gettoniera alle impostazioni di fabbrica.*

D) ***PER RIPRISTINARE LE IMPOSTAZIONI DI FABBRICA***:

1. A macchina spenta porre gli Switch del banco SW1 e quelli del banco SW2 in posizione OFF
2. accendere l'alimentazione
3. **Banco-switch SW1-** *Posizionare tutti i dip-switch su ON.*
4. Introdurre una dopo l'altra 3 monete di qualunque valore; dopo la prima, la spia rossa lampeggia, dopo la terza la bobina della gettoniera si attiva per due volte consecutive (doppio scatto), e la spia verde lampeggia a conferma dell'accettazione del ripristino.
5. La gettoniera esce automaticamente dalla modalità di ripristino/programmazione.

N.B.: nell'impostazione di fabbrica sono programmati i soli canali 1, 2 e 3, nel modo seguente:
canale 1 = 0,50 €
canale 2 = 1,00 €
canale 3 = 2,00 €
canali 4 e successivi = disponibili per programmazione, da realizzare con le modalità manuali viste al punto A), oppure via PC tramite il software AL66 WinProg Alberici.

ISTRUZIONI DI AUTOPROGRAMMAZIONE AL55 STD
Valide a partire dalla versione 1.5.0

A) *PER ESEGUIRE LA PROGRAMMAZIONE*:

1. Togliere l'alimentazione alla gettoniera

2. porre gli Switch del banco SW1 in posizione OFF

3. alimentare la gettoniera

4. **Banco SW1-** porre in ON il dip-switch corrispondente all'uscita da programmare e lasciare tutti gli altri in OFF. Ad es., l'uscita 2 richiede DS 2 in posizione ON, tutti gli altri in OFF

5. Introdurre una dopo l'altra 15 monete: la bobina della gettoniera si attiva per due volte consecutive (doppio scatto), a conferma dell'accettazione della programmazione.

6. Riportare gli switch in posizione OFF.

7. Spegnere l'alimentazione.

8. Per programmare altre monete in altri canali, ripetere le operazioni dal punto 2 al punto 5.

9. Per riavviare: porre in posizione ON i dip-switch del banco SW1 corrispondenti alle monete da accettare, e riaccendere l'alimentazione.

B) *PER COLLAUDARE LA PROGRAMMAZIONE ESEGUITA*:

1. Porre in ON tutti i dip-switch del banco SW1.
2. Accendere l'alimentazione e verificare l'accettazione o il rifiuto delle monete programmate.

C) ***PER RIPRISTINARE LE IMPOSTAZIONI DI FABBRICA***:

1. Spegnere l'alimentazione
2. Porre tutti gli Switch del banco SW1 in posizione OFF
3. Accendere l'alimentazione
4. **Banco-switch SW1-** *Posizionare tutti i dip-switch su ON.*
5. Introdurre una dopo l'altra 3 monete di qualunque valore; dopo la terza, la bobina della gettoniera si attiva a conferma dell'esecuzione del ripristino.
6. La gettoniera esce automaticamente dalla modalità di ripristino/programmazione.

N.B.: nell'impostazione di fabbrica sono programmati i soli canali 1, 2 e 3, nel modo seguente:
canale 1 = 2,00 €
canale 2 = 1,00 €
canale 3 = 0,50 €
canali 4 e successivi = disponibili per programmazione, da realizzare con le modalità manuali viste al punto A), oppure via PC tramite il software AL55 WinProg Alberici.

D) ***PER CANCELLARE TUTTI I CANALI***:

1. Spegnere l'alimentazione
2. Porre tutti gli Switch del banco SW1 in posizione OFF
3. Accendere l'alimentazione
4. **Banco-switch SW1-** *Lasciare tutti i dip-switch su OFF.*
5. Introdurre una dopo l'altra 3 monete di qualunque valore; dopo la terza, la bobina della gettoniera si attiva a conferma dell'avvenuta cancellazione.
6. La gettoniera esce automaticamente dalla modalità di ripristino/programmazione.

N.B.: una volta cancellato un canale, non è più possibile ri-attivarlo o ri-programmarlo manualmente: è necessario utilizzare il software di programmazione AL55 WinProg Alberici. E' però possibile riportare la gettoniera alle impostazioni di fabbrica.

INIBIZIONE MONETE :

METTERE IN OFF IL DIP SWITCH CORRISPONDENTE ALL'USCITA ASSEGNATA ALLA MONETA DA INIBIRE.
PER VERIFICARE LA CORRISPONDENZA FRA LA MONETA E L'USCITA, CONSULTARE LA COLONNA "OUT" PRESENTE NELLA GRIGLIA DELL'ETICHETTA.

• ***ESEMPIO:*** *per inibire le monete da un 1 euro programmate sull' OUT5 è necessario mettere in OFF il dip-switch 5. Invece per inibire le monete da 2 euro programmate sull' OUT 6 mettere in OFF il dip-switch*

5.3 Separazione monete

Tramite il software AL66 è possible definire con semplicità le direzioni per ogni moneta accettata, una volta selezionato il tipo di separatore. Le uscite usate per le bobine del separatore non possono essere usate allo stesso tempo per altre funzioni.
Vengono normalmente usate le uscite 5 (pin 3) e 6 (pin 4) sul connettore 10-p. Se il separatore è a 5 vie e quindi ha una terza bobina, si usa anche l'uscita 4 (pin 10).

I separatori ALBERICI sono:
- 3-vie NS3
- 3-vie NL3
- 3-vie VARIANT sovrapposto
- 3-vie VARIANT lineare
- 5-vie VARIANT

5.4 Display (solo AL66)

La gettoniera AL66 può pilotare i seguenti display ALBERICI:
SCHED670 SERIALE LCD 1x8 / **SCHED071** PARALLELO 2X16
A seconda della configurazione del selettore, il display può visualizzare:
- credito accumulato
- valore introdotto o resto
- tempo trascorso
- messaggi d'errore

Nota: non si può visualizzare allo stesso tempo il tempo e il valore introdotto o il resto.

5.5 Gestione del resto

L'utente può cancellare il resto residuo dopo l'erogazione del servizio (settare il tempo di conservazione a 0), mantenerlo in permanenza fino allo spegnimento (è la condizione di default: settare a 255), o cancellarlo dopo un intervallo (settabile da 1 a 254 secondi).

5.5.Z Gestione del resto Comma 6A Norme AAMS

Le norme emanate dai Monopoli di stato prevedono che le macchine da gioco con premio possano restituire l'eventuale resto di 0,50 € se entro un certo tempo non vengono introdotti gli 0,50 € mancanti al costo della singola partita, oppure a richiesta del giocatore. Questo è ottenibile in due modi:
- usando il separatore AVANT MH con la gettoniera AL66VC cctalk (dalla versione 1.3.0 in poi) con dip-switch 1 del banco SW2 in posizione OFF,
- usando il separatore AVANT 4l con la gettoniera AL66VCZ cctalk

Dal punto di vista dei comandi cctalk, la gettoniera AL66VCZ differisce dall'altra per la risposta all'Header 244 di richiesta codice prodotto: la AL66VC risponde "AL66V-c", invece la AL66VCZ risponde "AL66V-z".

5.6 Inibizione frode

Per incrementare l'effetto di protezione dei vari dispositivi meccanici (taglia-fili e anti-ripescaggio) ed elettronici incorporati (sensori di posizione delle monete e controllo di flusso), si può bloccare l'accettazione per un periodo programmabile (da o a 254 sec). Se il tempo viene settato a 255, la moneta successiva viene sistematicamente rifiutata.

6. Serial cctalk communication (English language)

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[4] in smal network.
Protocol is mostly used in gaming and casino machines but it can be implemented in any other tipes of machines that use same type of devices.
It is public protocol and free to use. Find documentation on: www.cctalk.org.

Communication protocol of ALBERICI coin selectors AL55/66 complies with generic specification 4.4

6.1 Communication specifications

Serial communication was derivated from RS232 standard.
It is low data rate NRZ (***N**on **R**eturn to **Z**ero*) asyncronous 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 suported.
Mesage integrity is controled by means of checksum calculation.

6.1.1 Baud rate

The baud rate of 9600 was chosen as compromise betwen 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*).

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 appirance see image1.

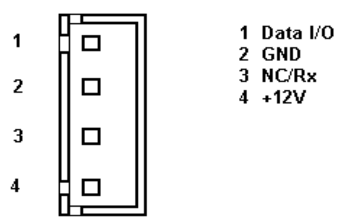


Image 6.1 communication connector

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

[4]Coin selectors, Hoppers(*pay out device*), Banknote readers etc.

6.2 Message structure

Each communication sequence consists of two mesage string.
Mesage string in case of simple checksum use is structured as folows:

[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[5]. 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

[5] For details see cctalk44-2.pdf, Address poll

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**.

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 mesage, and total lenght of message string will be 5 bytes. Although theoretically it will be possible to send 255 bytes of data becouse of some limitations in small micro controlers the number is limitet to 252[6].

6.2.3 Command headers (*Instructions*)

Total amount of possible cctalk command header is 255 with possibility to add sub-heaers using headers 100, 101, 102 and 103.
Header 0 stands for **ACK** (***ack****nowledge*) replay of device to host.
Header 5 stands for **NAK** (***N****o* ***ack****nowledge*) replay of device to host.
Header 6 is **BUSY** replay of device to host.
In all three cases no data bytes are transfered. Use of ACK and NAK headers are explained later on, for each specific mesage transfer.
Commands are devided in to several groups acording to aplication specifics:

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

ALBERICI Coin selectors AL55/66 use total of 55[7] instructions-headers.

6.2.4 Data

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

6.2.5 Checksum

Mesage 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 mesage.
If mesage is recieved and the addition of all bytes are non-zero then an error has occurred[8].
For noisy enviroment or higher security aplication it is possible to use more complex, 16 bit CRC CCITT checksum based on a polynomial of:
$\mathbf{x^{16} + x^{12} + x^{5} + 1}$ and initial value of CRC register **0x0000**.

Coin selectors AL55/66 are using simple checksum, but they could be set to operate with CRC-16 checksum on customer demand.

---

[6] 252 bytes of data, source address, header and checksum (total of 255 bytes)
[7] Some type of coin selectors do not support all headers
[8] See Error handling

6.3 Timing specification

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

6.3.1 Time beetwen two bytes

When reciving bytes within a mesage string, the comunication 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 recieve next mesage. The interbyte delay during transmition should be ideally **less than 2 ms** and **not greater than 10 ms**.

6.3.2 Time beetwen comand and reply

The time beetwen comand and reply is dependent on aplication specific for each comand. Some comands return data imediatly, and maximum time delay should be within **10 ms**. Others comands that must activate some actions in device, may return reply after the action is finished[9].

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.

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
255	FF	Factory specific test	Supported
254	FE	Simple poll	Return ACK
253	FD	Address poll	MDCES support
252	FC	Address clash	MDCES support
251	FB	Address change	MDCES support, non volatile
250	FA	Address random	MDCES support, non volatile
249	F9	Request polling priority	[02][32] 100x50=500 ms

[9] I.e. more than 100 mili sec for solenoid testing.

248	F8	Request status	[00] Ok
246	F6	Request manufacturer id	'Alberici'
245	F5	Request equipment category id	'Coin Acceptor'
244	F4	Request product code	'ALNNxn' NN=55/66, x=V/I/K, n=1-3 / Z
243	F3	Request database version	[01] remote file programming
242	F2	Request serial number	From 0 to 16.777.215
241	F1	Request software revision	'u3.n p3.m' n=0..9, m=0..9
240	F0	Test solenoids	Coil on for 100 ms
238	EE	Test output lines	Supported
237	ED	Read input lines	[In1=MSb,DIP-sw1][In2=MSb,DIP-sw2]
236	EC	Read opto states	bit0=opto1, bit1=opto2
233	E9	Latch output lines	Supported
232	E8	Perform self test	Supported
231	E7	Modify inhibit status	[inhibit 1][inhibit 2] total 16 position, volat.
230	E6	Request inhibit status	Supported
229	E5	Read buffered cred. or error c.	Five two byte event buffer
228	E4	Modify master inhibit status	bit0=0 inhibited ..1=enable, volatile
227	E3	Request master inhibit status	Supported
226	E2	Request insertion counter	[Rjct1-MSB][Rjct2][Rjct3-LSB]
225	E1	Request acceptance counter	[Rjct1-MSB][Rjct2][Rjct3-LSB]
221	DD	Request sorter override status	[FF] Normal sorting
219	DB	Enter new PIN number	Supported, non volatile
218	DA	Enter PIN number	ACK return if PIN is correct
216	D8	Request data storage availability	[00][00][00][00][00], not available
215	D7	Read data block	For encrypted data exchange!
214	D6	Write data block	For encrypted data exchange!
213	D5	Request option flags	bit0=0 cred. code format position
210	D2	Modify sorter paths	[coin pos][path], volatile
209	D1	Request sorter paths	Supported
202	CA	Teach mode control	Supported
201	C9	Request teach status	Supported
197	C5	Calculate ROM checksum	[ROM-H][ROM-L][EEPR-H][EEPR-L]
196	C4	Request creation date	Supported
195	C3	Request last modification date	Supported
194	C2	Request reject counter	[Rjct1-MSB][Rjct2][Rjct3-LSB]
193	C1	Request fraud counter	[Frd1-MSB][Frd2][Frd3-LSB]
192	C0	Request build code	'AL66 V1.0'
188	BC	Request default sorter path	[01] No sorting
185	B9	Modify coin id	Supported
184	B8	Request coin id	Supported
176	B0	Request alarm counter	Supported, one byte cumulative count
173	AD	Request thermistor reading	If thermistor is mounted
170	AA	Request base year	'2000'
169	A9	Request address mode	[84] addr. change non volatile(FLASH)
4	04	Request comms revision	[02][04][02] ,level2, isue4.2
3	03	Clear comms status variables	Supported
2	02	Request comms status variables	[Rx timeout][Rx b. ignored][Rx bad chks.]
1	01	Reset device	Software reset

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 mesage

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 **AL06V-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]**

6.5.1.9 Command 169 [hexA9], Request address mode

Coin selector respond with one byte data[10] information about addressing mode. Address could be stored in different type of memory(*RAM. ROM or EEPROM*), set with DIP-switch at printed circuit board or hard-wired at external connectors. Some devices support address change wit MDCES command headers[11]. Message format is:

Host sends: **[Dir] [00] [01] [A9] [Chk]**

Coin s. respond: **[01] [01] [Dir] [00] [Address mode] [Chk]**

ALBERICI coin selector has address is stored in non-volatile memory(*FLASH*) and address change is supported.

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

Host sends: **[02] [00] [01] [A9] [54]**

Coin s. respond: **[01] [01] [02] [00] [84] [78]**

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 mesage

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 mesage

---

[10] Details of description see in public document cctalk44-2.pdf

[11] Address change, Address random

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[12] 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 mesage

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 mesage

Host software must wait at least **100 ms**, to continue comunication 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[13].
Some commands are shared with other device like banknote reader or payout device.

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[14]. Coin selectors due to differences in mechanical and electrical construction has different acceptance speed. All events are registered in memory buffer with limited size[15] .
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[16].
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.

---

[12] Number of receive buffer overflow bytes.
[13] Sorter control commands
[14] Reading memory buffer from coin selector
[15] Five stage double byte memory buffer
[16] Dependant on mechanical type of coin selector (K, S type is faster) and coin

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[17].
Coin selector respond to command with two bytes of data. First byte is poll time unit and second is polling time value[18]. 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 x 50 = 500 ms

6.5.2.2 Command 248 [hexF8], Request Status

ALBERICI coin selectors has no additional COS[19] 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]**

6.5.2.3 Command 243 [hexF3], Request data base version

The respond to that command is version of coin data base. Version of data base is important for coin selectors with remote programming support.
For all ALBERICI coin selectors type AL55/66 current data base version is 00.
Message format is:

Host sends: **[Dir] [00] [01] [F3] [Chk]**

Coin s. respond: **[01] [01] [Dir] [Ver.] [Chk]**

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

Host sends: **[02] [00] [01] [F3] [0A]**

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

---

[17] For coin selector with respond time 2 ms and byte gap 1 ms it is 38 ms
[18] For details see, cctalk44-2.pdf
[19] **C**oin **O**n **S**tring

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.

6.5.2.5 Command 238 [hexEE], Test output lines

Host sends one byte mask to determinate which output line must be tested.
Coin selector output line that correspond to bit set in the mask will be pulsed for 100 ms and after that, ACK message will be transmitted. Message format is:

Host sends: **[Dir] [01] [01] [EE] [Mask.] [Chk]**

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

Example of message string for coin selector(*address 2*) first output(*pin7*) is:

Host sends: **[02] [01] [01] [EE] [01] [0D]**

Coin s. respond: **[01] [00] [02] [00] [FD]** Single pulse out 1 -> 100 ms, ACK

Bit positions for output test are:
- bit 0 Output 1(*pin 7*)
- bit 1 Output 2(*pin 8*)
- bit 2 Output 3(*pin 9*)
- bit 3 Output 4(*pin 10*)
- bit 4 Output 5(*pin 3*)
- bit 5 Output 6(*pin 4*)
- bit 6 Output 7(*pin 5*)
- bit 7 Not used

Unused output (*not programmed*) will not be turned on, but message ACK will be returned.

6.5.2.6 Command 237 [hexED], Read input lines

Coin selector respond with two data byte representing the state of DIP-switches and state of inputs In1(*pin 6*) and In2(*pin 5*)[20].
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, wile 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]**

[20] If In2 is programmed as input

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[21] 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 AL66V**2** 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]**

6.5.2.8 Command 233 [hexE9], Latch output lines

This instruction is similar to instruction 238. Host sends one byte mask to determinate which output line must be activated(*latch*). ACK message will be transmitted immediate. Coin selector output line that correspond to bit set in the mask will be latched and active till reset or new instruction with bit cleared is sent. Message format is:

Host sends: **[Dir] [01] [01] [E9] [Mask.] [Chk]**

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

Example of message string for coin selector(*address 2*) first output(*pin7*) is:

Host sends: **[02] [01] [01] [E9] [01] [12]**

Coin s. respond: **[01] [00] [02] [00] [FD]** Latch out 1 -> ACK

6.5.2.9 Command 232 [hexE8], Perform self-test

IMPORTANT WARNING: send only after at least 2 seconds from last power-on or from last reset command have elapsed.

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*).

[21] In some case group could contain more than one opto pairs

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

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

Ex. of message string for coin selector (*addr. 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 do 15 determinate coin positions from 1 to 16[22].
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!

[22] Positions are sent by coin selector during reading credit buffer or error codes (*header 229*)

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**[23] after power-up or reset is:

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

Coin s. respond: **[01] [02] [02] [00] [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[24] 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.

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

[23]Coin selector for Italian gambling machines
[24] If coin selector use CVF (*Coin Value Format*)

Coin selectors also use one eight bit counter[25] 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[26]. 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][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][00] [EE]**

First event, coin possition 1, sorter path 2 accepted

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

Second event, coin possition 2, sorter path 1 accepted

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

[25] Event counter
[26] 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[27] 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] [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[28]. 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] [00] [FA]**

6.5.2.17 Command 221 [hexDD], Request sorter override status

Coin selectors AL55/66 do not support override of sorter path.
Coin selector respond will be always: hex**[FF]** – Normal sorting.
Example of message string for coin selector(*address 2*) is:

Host sends: **[02] [00] [01] [DD] [20]**

Coin s. respond: **[01] [01] [02] [00] [FF] [FD]**

---

[27] Using command 210, Modify master inhibit status
[28] Credit sensor

6.5.2.18 Command 219 [hexDB], Enter new PIN number

Host send four byte data of new PIN number. If correct PIN was previously received[29] coin selector will accept the new PIN and respond with ACK message Coin selectors AL06x-c has PIN number stored in EEPROM. Message format is:

Host sends: **[Dir] [04] [01] [DB] [PIN1-LSB][PIN2][PIN3][PIN4-MSB] [Chk]**

Coin s. respond: **[01] [00] [02] [00] [FD]** ACK if PIN is correct

Coin s. respond: no respond if PIN is incorrect or not recieved

Example of message string for coin selector(*address 2*) with default PIN: hex**[00][00][00][00]** previously received and NEW pin hex**[01][02][03][04]** is:

Host sends: **[02] [04] [01] [DB] [01][02][03][04] [14]**

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

6.5.2.19 Command 218 [hexDA], Enter PIN number

Host send four byte data of PIN number. If PIN is correct, coin selector will respond immediately with ACK message. If PIN is incorrect the NAK message will be sent with time delay of 100 ms. PIN number of AL06x-c is stored in EEPROM. Message format is:

Host sends: **[Dir] [04] [01] [DA] [PIN1-LSB][PIN2][PIN3][PIN4-MSB] [Chk]**

Coin s. respond: **[01] [00] [Dir] [00] [Chk]** ACK if PIN is correct

Coin s. respond: **[01] [00] [Dir] [05] [Chk]** dly 100 ms ->NAK if PIN is incorrect

Example of message string for coin selector(*address 2*) with default PIN: hex**[00][00][00][00]** and wrong pin is:

Host sends: **[02] [04] [01] [DA] [01][00][00][00] [1F]**

Coin s. respond: **[01] [00] [02] [05] [F8]** dly 100 ms ->NAK if PIN is incorrect

6.5.2.20 Command 216 [hexD8], Request data storage availability

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

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

Coin s. respond: **[01] [05] [Dir] [00] [d1][d2][d3][d4][d5] [Chk]**

ALBERICI coin selectors AL55/66 currently do not support write or read to host accessible memory. Respond to command will be always as in example:

Host sends: **[02] [00] [01] [D8] [25]**

Coin s. respond: **[01] [05] [02] [00] [00][00][00][00][00] [F8]**

6.5.2.21 Command 213 [hexD5], Request option flags

Coin selector respond with one byte of data that describes type of coin format. For CVF (***C**oin **V**alue **F**ormat*) bit 0 is set to 1, and for coin position format value is "0". Other bits are not used and read always as "0". Message format is:

Host sends: **[Dir] [00] [01] [D5] [Chk]**

Coin s. respond: **[01] [01] [Dir] [00] [d1] [Chk]**

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

Host sends: **[02] [00] [01] [D5] [28]**

Coin s. respond: **[01] [01] [02] [00] [00] [FC]** Coin position format

[29] See next chapter

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*). Firs 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[30] , but recently 5 way sorters[31] 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. Example 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

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[32] , the coin selector will respond with message NAK !

[30] Maximum current consumption for each coil is 500 mA
[31] 5-way VARIANT sorter from ALBERICI
[32] Position bigger than 16

6.5.2.24 Command 202 [hexCA], Teach mode control

This command is used to start teach process(program coin recognition data).
The respond of coin selector is ACK if teach mode is supported.
With command header host must send number of channel to program.
ALBERICI coin selectors AL55/66 has possibility to program 16 different coins.
Some coin selectors that must not be reprogrammed for security or any other reason do not support this instruction[33]. If teach mode instruction is not supported coin selector will not respond to this instruction. Message format is:

Host sends: **[Dir] [01] [01] [C9] [ch][Chk]**
Coin s. respond: **[01] [00] [Dir] [00] [Chk]** ACK if ch is betwen 1-16
Coin s. respond: **[01] [00] [Dir] [05] [Chk]** NAK if ch is 0 or bigger than 16

Example of message string for coin selector(*address 2*) to teach (*program*) coin on position (*channel*) 1:

Host sends: **[02] [01] [01] [C9] [01] [31]**
Coin s. respond: **[00] [02] [00] [FD]** ACK

6.5.2.25 Command 201 [hexC9], Request teach status

This command is used during teach process, after instruction 202 Teach mode control.
The respond of coin selector is according to teach state.
There is two different format for this instruction.
Format "a" is with data hex**[00]**, after which coin selector respond with number of inserted coins and state of teach process.
Second format is "b" with data hex**[01]**, after which coin selector abort the teach process and respond with code dec**[252]**, teach aborted and number of inserted coins. Teach status codes are:

252	Teach aborted	253	Teach error
254	Teaching in progress(busy)	255	Teach completed

Message format (a) is:

Host sends: **[Dir] [01] [01] [C9] [00][Chk]** Request status

Message format (b) is:

Host sends: **[Dir] [01] [01] [C9] [01][Chk]** Abort teach process
Coin s. respond: **[01] [02] [Dir] [00] [coin nr.][status] [Chk]**

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 (**R**elative **T**o **B**ase **Y**ear)* [34] 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.

---

[33] Italian cctalk coin selectors AL06V-c for gambling machines
[34] For details see cctalk protocol, document cctalk44-2.pdf

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[35]. 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]**

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[36]. 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]**

6.5.2.31 Command 185 [hexB9], Modify coin ID

With this command it is possible to change coin ID message string that will be used in communication with host. String has 6 ASCII characters:

- Country code(*2 bytes*)
- Value(*3 bytes*)
- Mint issue(*1 byte*)

Each country has a 2 letter designator code described in ISO 3166-1-A2.
ALBERICI coin selectors has limited possibility to change the country code.

[35] Up-grade of FLASH program memory

[36] Coins with close recognition parameters sometime called "killer coin or channel"

Table of four country codes must be programmed first(*by customer*). Code sent by the host must be one from the table or NAK message will be returned to host.
Default country code table programming for europe is:

EU	07B2 + 07B3
TK	07B4 + 07B5
GB	07B6 + 07B7
…	07B8 + 07B9

"EU" is code for euro coins, "TK" is for token, "GB" is for british pounds and ".." is code for non programmed coin positions.
It is possible to change or add country code by writing code in memory location from address hex 07B2 (*4x2 bytes*):

slave+numbytes+master+"FF 03"+"07 B6"+Hex(asci code)+Hex(asci code)+cks
- Example: 02+05+01+FF+03+07+B6+47+42+cks
ACK message will be returned.

Now coin ID message string can be modified:
slave+numbytes+master+Header B9+coin position+CountryCode1+CountryCode2+chr(table 6.5)1+chr(table 6.5)2+chr(table 6.5)3+chr(Mint)+cks
- Example:
02+07+01+B9+01+47+42+32+30+30+41+cks
ACK message will be returned.

Coin value code must be selected from table 6.5 or else NAK message will be returned to host.

Mint issue must also be selected as "A", "B", "C" or "." for non programmed coins!

3 x ASCII Characters	Value
001	1
002	2
2.5	2,5
005	5
010	10
020	20
025	25
050	50
100	100
200	200
250	250
500	500
…	Not prog.

Table 6.5 Coin value codes

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 programed

6.5.2.33 Command 176 [hexB0], Request alarm counter

Coin selector respond with one bytes of alarm counter data.
Alarm counter is set to zero after power up or reset command. It is incremented each time a coin acceptor detect any type of erroneous coin acceptance[37].

[37] Alarms: Coin direction error(Jojo), coin to slow(Coin jam) or coin to fast

Message format is:
Host sends: **[Dir] [00] [00] [B0] [Chk]**
Coin s. respond: **[01] [01] [Dir] [00] [Cunt] [Chk]**

Example of message string for coin selector(*address 2*) after power-up is:
Host sends: **[02] [00] [01] [B0] [4D]**
Coin s. respond: **[01] [03] [02] [00] [00] [FC]**

6.5.2.34 Command 173 [hexAD], Request thermistor reading

Some coin selectors AL66[38] has built in linear temperature sensor.
Using this command is possible to read temperature on surface of coin selector PCB.
If temperature sensor is not built in coin selector will not respond to this command.
Temperature sensor is linear type, with 1 unit change for one degree Celsius change.
For 0°C value will be dec**[50]**, for ie. 25°C it will be dec**[75]**, for -10°C it will be dec**[40]** and for 50°C it will be dec**[100]**. Message format is:
Host sends: **[Dir] [00] [00] [AD] [Chk]**
Coin s. respond: **[01] [01] [Dir] [00] [Temp] [Chk]**

Example of message string for coin selector(*address 2*) at ambient temperature of 25°C is:
Host sends: **[02] [00] [01] [AD] [50]**
Coin s. respond: **[01] [01] [02] [00] [4B] [B1]**

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 !

[38] Coin selectors for use in extreme ambient temperature conditions(external use)

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]** Brodcast mesage

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

6.5.3.3 Command 251 [hexFB], Address change
Command Address change is issued to a specified device only. Coin selector respond with ACK message. Message format is:

Host sends: **[Dir] [01] [01] [FB] [Address] [Chk]**

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

Example of message string for coin selector(*address 2*) change to address to 20:

Host sends: **[02] [01] [01] [FB] [14] [ED]**

Coin s. respond: **[01] [00] [02] [00] [FD]** ACK Address is now 20

Coin selector does not respond to attempt of change an address to 0 or 1.

6.5.3.4 Command 250 [hexFA], Address random
Command Address random has the same respond from coin selector. New address is not sent because each device set its own random address.
Host software sometime can issue this command as broadcast. This will cause change of all device addresses. Coin selector respond with ACK message. Message format is:

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

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

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

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

Coin s. respond: **[01] [00] [02] [00] [FD]** ACK Address is changed

Example of broadcast message string for coin selector is:

Host sends: **[00] [00] [01] [FA] [05]** Brodcast mesage

Coin s. respond: **[01] [00] [00] [00] [FD]** ACK Address is changed

Coin selector has internal mechanism that prevent setting of address 0 or 1!

6.5.4 ALBERICI specific command

First command header for specific factory setting and testing is **255 Factory set-up and test**. This command has several modes of use and some of them are factory secret and are not explained in this document.
Beside this next chapter describe two modified cctalk commands[39] for encrypted exchange of data between host and coin selector.

6.5.4.1 Command 255, Factory set-up and test

This instruction header is used only for factory testing and programing!
With instruction command header one data byte must be sent for definition of command mode. Modes are:

- Mode 0(*hex 00*) No description – reserved for factory use only
- Mode 1(*hex 01*) No description – reserved for factory use only
- Mode 2(*hex 02*) Read coin selector memory data
- Mode 3(*hex 03*) Write coin selector memory data
- Mode 4(*hex 04*) Analog circuit test
- Mode 5(*hex 05*) Coin parameter test
- Mode 6(*hex 06*) Up-grade FLASH(*program memory*)
- Mode 7(*hex 07*) Re-initialization

First two modes reserved for factory use - not described in this document!

To use read and write instruction user must have basic knowledge about coin selectors memory data organization[40]!

6.5.4.1.1 Command 255 mode 2, Read coin sel. memory

This instruction sends back to host block of memory data that was requested.
Data memory of coin selector is divided into a 6 group:

- Coin channel data address hex 0600 do 06FF(*256 byte-a*)
- Input/Output data address hex 0700 do 073F(*64 byte-a*)
- Factory common data address hex 0740 do 079F(*96 byte-a*)
- User general data address hex 07A0 do 07E9(*74 byte-a*)
- Statistic setting address hex 07EA do 07FD(*20 byte-a*)
- Statistic counters address hex 0800 do 083F(*64 byte-a*)

Factory key or user PIN protection are some time set to disable the access to some blocks of memory. Usually all memory is accessible for red instruction.
On any attempt to read memory location that is protected without clearing key or PIN protection mechanism, coin selector will respond with **NAK** message.
If number of blocks to read exceed block range, coin selector will also respond with **NAK** message. Maximum memory block read of 137 data bytes are limited by coin selector transmit buffer that has 142 bytes[41]. If block size extend this number coin selector will respond with NAK message to. Message string format is:

---

[39] Commands 214 Write data block and 215 Read data block
[40] Details are available on customer request, see document AL55/66-MemDataOrg-v1.pdf
[41] Destination address+Bytes nr.+Source address+Header+59 data+Checksum=142

Host sends: **[Dir] [04] [01] [FF] [02][Start add-hi][Start add-lo][data nr.] [Chk]**
Coin s. respond: **[01] [n] [Dir] [00] [data 1] [data 2] . . . [data n] [Chk]**

Start add-hi is start address hi byte, while start add-lo is address low byte. Data nr. represent the number of data to read (block size).
Data 1 to data n are requested coin selector memory data.
The example of message string for reading of first coin channel data is:
Host sends: **[02] [04] [01] [FF][02] [06][00] [10] [E2]**
Coin s. respond: **[01][10][02][00][89][87][B4][77][A7][9F][08][08][0C][16][0A][04][08][01][09][C8][52]**

6.5.4.1.2 Command 255 mode 3, Write coin sel. memory

This command sends block of data to be written in the coin selector RAM memory.
As for the previous command(*read*) access to some memory blocks could be protected by factory key or user PIM mechanisam. Factory common data are usually protected with factory key and statistic counters and setting with user PIN. Exeption is made for statistic countet write. Any attempt to write in to statistic counter will erase those couners, thus protecting user from manipulation with statistics!
Number of bytes sent in command string are representing number of statistic to be erased. Message string format is:
Host sends: **[Dir] [n+3] [01] [FF] [03][Start add-hi][Start add-lo][data 1]. . . [data n] [Chk]**
Coin s. respond: **[01][00] [Dir] [00] [Chk]** ACK

The example of message string for programming first output of coin selector **AL55/66** (*address 2*) as pulse parallel output for coin(*channel*) position 1 is:
Host sends: **[02] [0B] [01] [FF][03] [07][00] [01][00][00][00][14][00] [D2]**
Coin s. respond: **[01][00][02][00] [FD]** ACK

The example of message string for erase of all 20 statistic counters[42] for coin selector **AL55/66** (*address 2*) is:
Host sends: **[02] [04] [01] [FF][02] [08][00] [14] [DB]**
Coin s. respond: **[01][00][02][00] [FD]** ACK

User PIN must be sent before erase(*write*) of statistic counters otherwise coin selector will respond with **NAK** message.

Data write will change data in coin selector RAM, and changes will be lost if power supply turns off! Use write to FLASH command to save changes[43]!

To save all coin selector RAM data in to FLASH, send write command with no data:
Host sends: **[02] [01] [01] [FF][03] [FA]**
Coin s. respond: **[01][00][02][00] [FD]** dly 20 -100 ms **->**ACK
Coin selector will reply with ACK if write to FLASH was successful.

6.5.4.1.3 Command 255 mode 4, Analog circuit test

This command returns to host string of data that can be used to test analog measuring circuit state. Details are described in internal documents and not available to users.

6.5.4.1.4 Command 255 mode 5, Coin parameter test

This command returns to host string of data from the last coin measurement.
This data are coin measured parameters and could be used to create and analyze coin data base. Message string format is:
Host sends: **[Dir] [01] [01] [FF][05] [Chk]**
Coin s. respond: **[01] [6] [Dir] [00] [data 1] [data 2] . . . [data 6] [Chk]**

[42] 20 x 3 byte of memory!
[43] Special case write command

Data string description:
- data 1 Measured coin parameter 1, AM1
- data 2 Measured coin parameter 2, PH1
- data 3 Measured coin parameter 3, AM2
- data 4 Measured coin parameter 4, PH2
- data 5 Measured coin parameter 5, PI3
- data 6 Measured coin parameter 6, DIM

The example of message string after insertion of one Euro coin for coin selector **AL66** (*address 2*) is:

Host sends: **[02] [01] [01] [FF][05] [F8]**
Coin s. respond: **[01] [6] [02] [00] [8A] [80][8C] [5C] [AB] [8A] [D0]**

Coin measured parameters(*data*) are available for read till next coin insertion.
This instruction is used by our coin programming software.

6.5.4.1.5 Command 255 mode 6, Up-grade FLASH

This command is used to up-grade coin selector FLASH program memory.
User can up-grade coin selector in cases when he knead some new function or improvement, which where not available at the moment when coin selector was purchased. This instruction is used by our coin selector programming software and generally is not useful to most users. Additional information are available on request.

Up-grade files are encrypted in factory and decryption is done internally by monitor program of coin selector.
Coin selector will accept only correct type of up-grade file!

6.5.4.1.6 Command 255 mode 7, Factory reset

This command will set all coin selector data to initial factory programming values!

Warning: This could lead to unwanted coin selector function if different coin , input/output or user setting data where programmed!

The example of message string reset the factory settings is:

Host sends: **[02] [01] [01] [FF][07] [F6]**
Coin s. respond: **[01][00][02][00] [FD]** ACK

After that, coin selector must be switched off/on!

7. Dati tecnici

Caratteristiche meccaniche	
Formato	3½" standard
Dimensioni	88 x 102 x 52 mm
Peso	200 g
Caratteristiche elettriche	
Tensione di alimentazione min.	8 V DC
Tensione di alimentazione max.	26 V DC
Assorbimento	
In accettazione	350 mA(*30 ms*)/100 mA
In misurazione	≤25 mA
In attesa (stand by)	≤20 mA
Risparmio energetico standard	≤2,5 mA
Autorisveglio	≤3,5 mA
Tipo uscita	Open collector Darlington
Tensione uscita di saturazione	≤1 V
Tensione uscita max.	50 V
Corrente uscita max.	250 mA
Tensione attivazione ingr. min.	3 V
Tensione ingresso max	50 V
Impedenza d'ingresso	≈55 kΩ
Accettazione monete	
Numero canali moneta	16
Diametro min. moneta	16 mm
Diametro max. moneta	32 mm
Spessore moneta	1 to 3,4 mm
Dati risposta	
Tempo di attivazione all'accensione	≤200 ms
Tempo di attivazione al risveglio	≤50 ms
Tolleranza impulso e time-put	± 2%
Condizioni ambientali	
Temperatura ambiente operativo	0°C to 60°C
Temperatura di magazzinamento	-30°C to 70°C
Umidità	fino a 75% *non condensata*
	fino a 95% (vers. tropicalizzata)
Compatibilità EMC	
Questo prodotto rispetta le normative EN55014-1 e EN55014-2	

Alberici®

Progettazione e produzione di sistemi di pagamento, accessori per videogames e macchine vending
Design and manufacture of payment systems, accessories for videogames and vending machines

Via Ca' Bianca 421
40024 Castel San Pietro
Terme (BO) – ITALY

Tel. + 39 051 944 300
Fax. + 39 051 944 594

http://www.alberici.net

info@alberici.net