Favero®
ELECTRONIC DESIGN

# PC-INTERFACE-32 (Art.745-4)
# PC-INTERFACE-64 (Art.745-6)
# PC-INTERFACE-96 (Art.745-8)



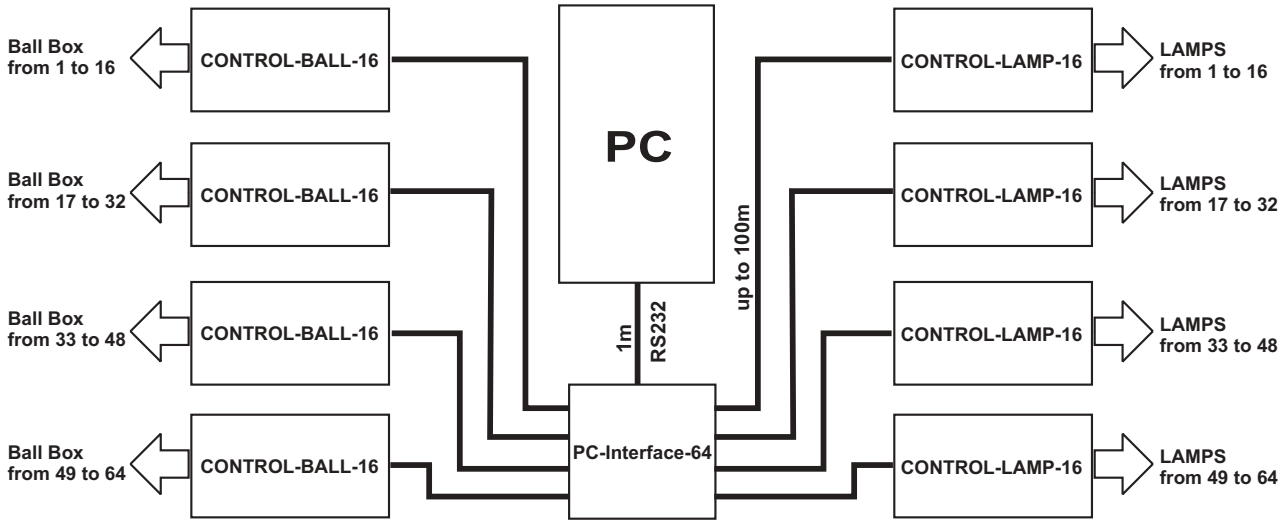

**FAVERO ELECTRONIC DESIGN**
Via R. Lombardi, 64 - 31030 Arcade (TV) - Italy
Tel. +39 0422 874140 Fax +39 0422 874141
http://www.favero.com - Email: favero@favero.com

# PC-INTERFACE

The **PC-Interface-96** is a device useful for those who wish to manage a billiards room with a PC (Personal Computer) using a own customized software application. Such device, connected to the personal computer RS232 serial port, allows the switching on and the shutdown of the billiards lamps and also the verification of the status of the ball boxes (open or closed) and their opening to distance.

The PC-Interface-96 may manage up to 96 billiards (96 lamps + 96 ball boxes) through the suitable controllers CONTROL-LAMP-16 and CONTROL-BALL-16.
To a PC-Interface-96 it is possible to connect up to 6 controllers for lamps CONTROL-LAMP-16 (6x16=96), and 6 controllers for ball boxes CONTROL-BALL-16 (6x16=96).

We manufacture also the PC-Interface-32 model, to which you can connect a maxim of 2 CONTROL-LAMP-16 controllers and 2 CONTROL-BALL-16 controllers to manage up to 32 billiards and the PC-Interface-64 model, to which you can connect a maxim of 4 CONTROL-LAMP-16 controllers and 4 CONTROL-BALL-16 controllers to manage up to 64 billiards.
For example here is a simplified diagram of the PC-Interface-64 connections:




The **PC-Interface-96** has a serial cable RS232 (with DB9 connector) 1 meter long for the direct connection to the PC. Every CONTROL-LAMP-16 or CONTROL-BALL-16 controller is provided of a cable 15 meters long for his direct connection to the PC-Interface-96. Such cable can be lengthened easily up to 100 meters (it is a standard 4 wires telephone cable).
All the PC-Interface-96 ports used for the connection to the controllers are of the **opto-coupled** type, so the PC is protected against electrical noise or overvoltages which could come from all the electric plant of the controllers.

**IMPORTANT NOTE:**
The PC-Interface-96 is provided with a CD containing a software component OCX (ActiveX control) with embedded communication protocol between the PC and the PC-Interface-96 . Anyone wants develop a program to manage lamps and ball boxes of the biliards through PC has to include such component in his application. Doing that the programmer will have the big advantage of don't have to be worried to develop any driver for the communication between the **PC-Interface-96** and the PC because it will be the component same to provide it.
A little example program is furthermore provided so that on the one hand, it allows to any user to quickly verify the working of the PC-Interface-96 and on the other it gives an example of as a programmer should use the above-mentioned component to create a software application. Both, OCX component and example program have been developed and tested with Microsoft Visual Basic 6.0 platform.

# ActiveX Component
## for software communication with the PC-Interface-96 device.

The ActiveX is a useful software component to develop customized application that have to communicate with the **PC-Interface-96** device to control lamps (CONTROL-LAMP) and balls (CONTROL-BALL) in the billiards rooms. By using methods and properties of this component it is possible get or put informations or perform commands without to be worried of the protocol of communication across the RS232 serial port.

## *Properties of the component:*

**ComPort [Read/Write]- data type: Byte.**
Using these properties it is possible to get or to set the numbers of serial port used by the component. The default value is 1.

**ComPortSettings [Read/Write]- data type: String**
Setup or returns the serial port communication parameter like speed, parity, bit and stop bit numbers. The default value is “9600,n,8,1” that means “9600” bps for speed, “n” (none) for parity, “8” for data bit and “1” for stop bit.
This parameters MUST NOT be changed because they are the only in order to operate correctly with the PC-Interface-96.

**BoxStatus [Read only]- data type: Object- BoxArray**
It returns the open/close status as a 96 elements matrix.
This property may be up-to-date by the ReadStatus method or setting up the UpdateTime property distinct from 0.

**ControlBallStatus [Read only]- data type: ControlBallArray**
It returns the status of the connected CONTROL-BALL-16 as an 8 elements matrix.
This property may be up-to-date by the ReadStatus method or setting up the UpdateTime property distinct from 0.

**TxtBoxStatus [Read only]- data type: String**
It returns the open/close status as a string of 96 elements .
The location of the character detects the box number, the value " 1 " or " 0 " announces if the box is open or closed. This property may be up-to-date by the ReadStatus method or setting up the UpdateTime property distinct from 0.

**TxtControlBallStatus [Read only]- data type: String**
It returns the status of the connected CONTROL-BALL-16 controllers as 8 characters string even if the maximum numbers of controllers on-line is 6. The location of the character detects the controller number, the value " 1 " or " 0 " announces if the box is open or closed. This property may be up-to-date by the ReadStatus method or setting up the UpdateTime property distinct from 0.

**UpdateTime [Read/Write]- data type: Byte**
It sets up or returns the updating time of the property over described. The value is express in seconds, if set to 0 the automatic updating comes disabled. In this case to update the BoxStatus, ControlBallStatus, txtBoxStatus, TxtControlBallStatus property is necessary recall the ReadStatus method. This property has to be set before recall the StartControl method. We suggest to set a value from 1 to 3 seconds in order to get a more rapid recognition of the closing/opening of the ball boxes.

## *Methods of the component:*

**StartControl- any parameter**
It actives the parameter.

**StopControl - any parameter**
It disables the parameter.

**ReadStatus - any parameter**
It forces the updating of the BoxStatus, ControlBallStatus, txtBoxStatus, TxtControlBallStatus properties. May be recalled at any time. The properties come update after about 0.4 seconds ( the response time of the interface ).

**OpenBox Parameter - BoxNumber (byte) from 1 to 96.**
It recalls the opening command of the ball box. It is necessary to specify the numbers of the box to open ( obligatory parameter ). If the box is opened correctly comes produced a BoxOpen event that allows to verify if the command has been approved or if for any mechanical or electric problem the command has not been performed.

**TurnOnLight- Parameter - LightNumber (byte) from 1 to 96.**
It changes the status to ON of the specified relay in the parameter and consequently it switches on the correspondent light.

**TurnOffLight- Parameter - LightNumber (byte) from 1 to 96.**
It changes the status to OFF of the specified relay in the parameter and consequently it switches off the correspondent light.

**TurnOnAllLight - any parameter**
It changes the status to ON of all the relays and consequently it turns on all the lights connected to them.

**TurnOffAllLight - any parameter**
It changes the status to OFF of all the relays and consequently it turns off all the lights connected to them.

## *Events of the component:*

**OpenBox - Parameter - BoxNumber (byte) from 1 to 96**
This event comes produced when the box that contains the balls comes opened. The event comes produced every time a ball box is opened, by manually or through the OpenBox method. The returned parameter announces the box number that has been opened.

**BoxClose - Parameter - BoxNumber (byte) from 1 to 96.**
This event comes produced when the box comes closed. The returned parameter announces the box number that has been closed. We suggest to set the UpdateTime property to a value from 1 to 3 seconds so the event happens within a short time.

**BoxStatusChange - Parameters: BoxNumber (byte) from 1 to 96. Status (byte) 0-1**
This event come produced when the status of the balls box is changed from open to closed or viceversa. The parameters return the numbers of the box that has changed status (BoxNumber) and the present status of the box (status). We suggest to set the UpdateTime property to a value from 1 to 3 seconds so that the opening or closing of a ball box is recorded within this time.

**ControlBallStatusChange - Parameter: Status ( String )**
This event comes produced when one or more on-line controllers (CONTROL-BALL) change status, for example come disconnected or they are fault. The status parameter is a string of 8 characters, the location of the character announces the number of the controller and the value " 1 " or " 0 " announces if it is on-line and working or not.

**ControlError - Parameters: StrErrorMessage (String)**

**ErrorNumber (Enumerator errors)**
This event comes produced when an error of communication is verified on the serial port or on the component it self. The StrErrorMessage parameter contains the error message, the ErrorNumber parameter the reference numbers of the error.

| Error Number | Enumerated constant |
|---|---|
| 1 | ComPortGenericError |
| 10 | ComPortOlreadyOpen |
| 11 | ComPortNotAvailable |
| 1001 | ComPortEventBreak |
| 1004 | ComPortEventFrame |
| 1006 | ComPortEventOverrun |
| 1008 | ComPortEventRxOver |
| 1009 | ComPortEventRxParity |
| 1010 | ComPortEventTxFull |
| 1011 | ComPortEventDCB |
| 10000 | GenericError |
| 10001 | StringNotValid |

# Constants and their values:

enmBoxStatus
BoxClosed = 0
BoxOpen = 1

enmControlBallStatus
Disconnected = 0
Connected = 1

BoxArray
Box1  Box96 (Byte)

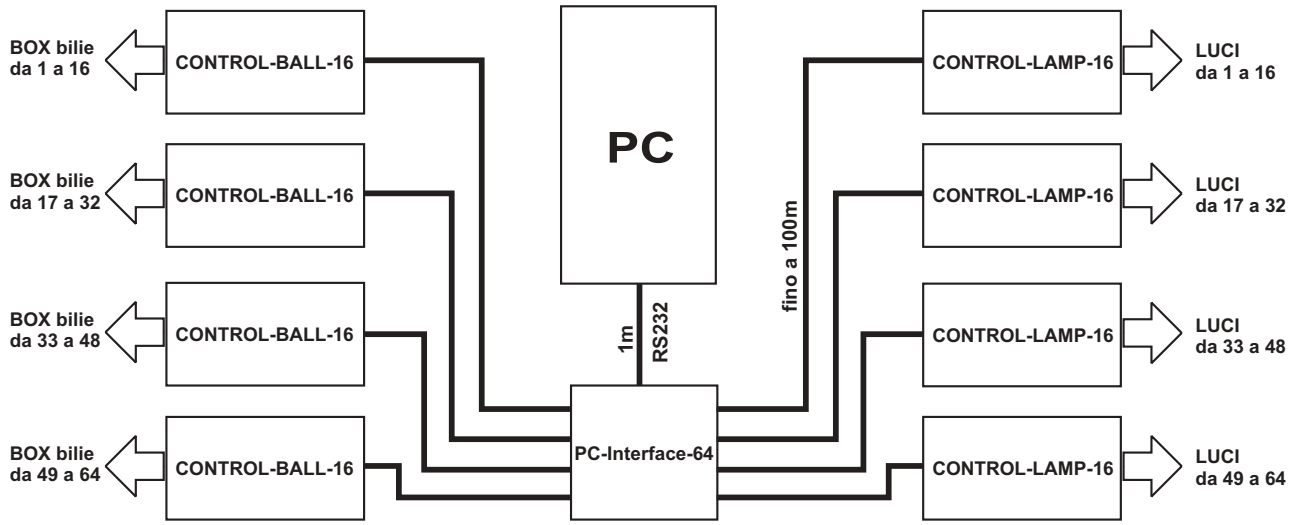
ControlBallArray
ControllBall1  ControllBall8 (Byte)

# PC-INTERFACE

Il **PC-Interface-96** è un dispositivo utile per coloro che desiderano gestire una sala biliardo con un PC utilizzando un proprio programma.
Tale dispositivo, che viene collegato ad una seriale RS232 del PC, permette l'accensione e lo spegnimento dei lampadari dei biliardi ed anche la verifica dello stato dei cassetti contenenti le bilie (aperti o chiusi) e la loro apertura a distanza.

Il dispositivo PC-Interface-96 può gestire fino a 96 biliardi (96 lampadari + 96 cassetti bilie) attraverso gli appositi controllori CONTROL-LAMP-16 e CONTROL-BALL-16.
Ad un PC-Interface-96 possono essere collegati fino a 6 controllori per le luci CONTROL-LAMP-16 (6x16=96), e 6 controllori per i cassetti bilie CONTROL-BALL-16 (6x16=96).

Vengono prodotti anche il modello PC-Interface-32 al quale è possibile collegare massimo 2 controllori CONTROL-LAMP-16 e 2 controllori cassetti CONTROL-BALL-16 per la gestione massima di 32 biliardi ed il modello PC-Interface-64 al quale è possibile collegare massimo 4 controllori CONTROL-LAMP-16 e 4 controllori cassetti CONTROL-BALL-16 per la gestione massima di 64 biliardi.
Ecco uno schema semplificato delle connessioni di un PC-Interface-64:




Il **PC-Interface-96** è fornito di cavo seriale RS232 (con connettore tipo DB9) lungo 1 metro per la connessione diretta al PC. Ogni controllore CONTROL-LAMP-16 o CONTROL-BALL-16 viene fornito di un cavo lungo 15 metri per la sua connessione diretta al PC-Interface-96. Tale cavo può essere facilmente allungato fino a 100m (trattasi di un comune cavo telefonico a 4 fili). Tutte le uscite del PC-Interface-96 utilizzate per il collegamento ai controllori sono del tipo **opto-isolate** e quindi il PC è protetto dai disturbi elettrici o sovratensioni che potrebbero provenire da tutto l'impianto elettrico dei controllori.

**NOTA IMPORTANTE:**
Il PC-Interface-96 viene fornito con un CD contenente un componente software OCX (controllo ActiveX) che contiene il protocollo di comunicazione tra il PC ed il PC-Interface-96. Chiunque voglia sviluppare un programma per controllare le luci ed i cassetti bilie tramite PC deve includere tale componente nella sua applicazione così facendo il programmatore avrà il grosso vantaggio di non doversi preoccupare di sviluppare alcun driver per far comunicare il PC-Interface-96 con il PC perché sarà il componente stesso a fornirglielo.
Viene inoltre fornito un piccolo programma software di esempio che, da una parte permette ad un qualsiasi utente di verificare velocemente il funzionamento del PC-Interface-96 e dall'altra costituisce un esempio di come poter usare il suddetto componente per creare un'applicazione.
Sia il componente OCX che il programma di esempio sono stati sviluppati e testati su piattaforma Microsoft Visual Basic 6.0.

# Controllo ActiveX
## per la comunicazione con l'interfaccia PC-Interface-96

Il controllo ActiveX è un oggetto utile per lo sviluppo di applicazioni personalizzate che devono comunicare con l'interfaccia PC-Interface-96 per il controllo lampade dei biliardi (CONTROL-LAMP) e dei cassetti delle bilie (CONTROL-BALL). Mediante metodi e proprietà del controllo è possibile ottenere o impostare informazioni o eseguire comandi senza doversi preoccupare del protocollo di comunicazione attraverso l'intefaccia Rs232.

## *Proprietà del controllo:*

**ComPort [Lettura/Scrittura] – Tipo di dati: Byte**
Mediante questa proprieta è possibile ottenere o impostare il numero di porta seriale utilizzata dal controllo. Il valore di default è 1.

**ComPortSettings [Lettura/Scrittura] – Tipo di dati: String**
Imposta o restituisce i parametri di comunicazione della porta seriale quali la velocità di comunicazione, il controllo di parità, il numero di bit dati ed il numero di stop bit. Il valore di default è "9600,n,8,1" cha sta ad indicare 9600 bps di velocità nessun controllo di parità, 8 bit di dati ed 1 stop bit.
Questi parametri non vanno modificati perché sono gli unici in grado di far funzionare l'interfaccia PC-Interface-96.

**BoxStatus [Sola Lettura] – Tipo di dati: Object - BoxArray**
Restituisce lo stato dei box aperti/chiusi sotto forma di matrice di 96 elementi. Questa proprietà è aggiornabile mediante il metodo ReadStatus oppure impostando la proprietà UpdateTime diversa da 0.

**ControlBallStatus [Sola Lettura] – Tipo di dati: Object - ControlBallArray**
Restituisce lo stato dei CONTROL-BALL-16 collegati sotto forma di matrice di 8 elementi. Questa proprietà è aggiornabile mediante il metodo ReadStatus oppure impostando la proprietà UpdateTime diversa da 0.

**txtBoxStatus [Sola Lettura] – Tipo di dati: String**
Restituisce lo stato dei box aperti/chiusi sotto forma di stringa di 96 caratteri. La posizione del carattere indica il numero del box, il valore "1" oppure "0" indica se il box è aperto oppure chiuso. Questa proprietà è aggiornabile mediante il metodo ReadStatus oppure impostando la proprietà UpdateTime diversa da 0.

**txtControlBallStatus [Sola Lettura] – Tipo di dati: String**
Restituisce lo stato dei ControlBall collegati sotto forma stringa di 8 caratteri anche se il massimo numero di controllori collegati è 6. La posizione del carattere indica il numero del controllore ed il valore "1" o "0" indica se il controllore è collegato correttamente oppure no. Questa proprietà è aggiornabile mediante il metodo ReadStatus oppure impostando la proprietà UpdateTime diversa da 0.

**UpdateTime [Lettura/Scrittura] – Tipo di dati: Byte**
Imposta o restituisce il tempo di aggiornamento delle proprietà sopra descritte. Il valore è espresso in secondi, se impostato a 0 l'aggiornamento automatico viene disattivato, in questo caso per aggiornare le proprietà BoxStatus, ControlBallStatus, txtBoxStatus, txtControlBallStatus e necessario richiamare il metodo ReadStatus. Questa proprietà deve essere impostata prima di richiamare il metodo StartControl. Si consiglia di impostare un valore da 1 a 3 per ottenere un più rapido riconoscimento della chiusura/apertura dei cassetti.

## *Metodi del controllo:*

**StartControl – Nessun Parametro**
Attiva il controllo

**StopControl – Nessun Parametro**
Disattiva il controllo

**ReadStatus – Nessun Parametro**
Forza l'aggiornamento delle proprietà BoxStatus, ControlBallStatus, txtBoxStatus, txtControlBallStatus. Può essere richiamato in qualsiasi momento. Le proprietà vengono aggiornate dopo circa 0.4 secondi (il tempo di risposta dell'interfaccia).

**OpenBox – Parametro – BoxNumber (Byte) da 1 a 96**
Richiama il comando di apertura del box bilie. E' necessario specificare il numero del box da aprire (parametro obbligatorio). Se il box si apre correttamente viene generato un evento BoxOpen che ci permette di verificare se il comando è stato eseguito o se per qualsiasi motivo meccanico o elettrico ciò non è avvenuto.

**TurnOnLight – Parametro – LightNumber (Byte) da 1 a 96**
Cambia lo stato a ON del relè specificato nel parametro e di conseguenza accende la luce corrispondente.

**TurnOffLight – Parametro – LightNumber (Byte) da 1 a 96**
Cambia lo stato a OFF del relè specificato nel parametro e di conseguenza spegne la luce corrispondente.

**TurnOnAllLight – Nessun parametro**
Cambia lo stato a ON di tutti i relè e di conseguenza accende tutte le luci ad essi collegate.

**TurnOffAllLight – Nessun parametro**
Cambia lo stato a OFF di tutti i relè e di conseguenza spegne tutte le luci ad essi collegate.

## *Eventi del controllo:*

**BoxOpen – Parametro – BoxNumber (Byte) da 1 a 96**
Questo evento viene generato quando il box che contiene le bilie si apre. L'evento viene generato ad ogni apertura di un cassetto, sia che questa apertura sia avvenuta manualmente che tramite il metodo OpenBox. Il parametro restituito indica il numero del box che è stato aperto.

**BoxClose – Parametro – BoxNumber (Byte) da 1 a 96**
Questo evento viene generato quando il box viene chiuso. Il parametro restituito indica il numero del box che è stato chiuso. Si consiglia di settare la proprietà UpdateTime ad un valore da 1 a 3 secondi affinchè l'evento avvenga in tempi più rapidi.

**BoxStatusChange – Parametri: BoxNumber (Byte) da 1 a 96. Status (Byte) 0 – 1**
Evento generato quando lo stato dei cassetti portabilie cambia di stato, viene aperto o chiuso. I parametri restituiscono il numero del box che ha cambiato stato (BoxNumber) e lo stato attuale del box (Status). Suggeriamo di impostare la proprietà UpdateTime ad un valore da 1 a 3 secondi così che una apertura o chiusura di un cassetto bilie viene riconosciuta entro tale tempo.

**ControlBallStatusChange - Parametro: Status (Stringa)**
Questo evento viene generato quando uno o più controllori dei cassetti (CONTROL-BALL) collegati cambiano di stato ad esempio vengono scollegati o si guastano.
Il parametro Status è una stringa di 8 caratteri, la posizione del carattere indica il numero del controllore ed il valore "1" o "0" indica se è collegato e funzionante oppure se non lo è.

**ControlError – Parametri: strErrorMessage (Stringa)**
**ErrorNumber (Enumeratore errori)**
Questo evento viene generato quando si verifica un errore sia di comunicazione sulla porta seriale che sul controllo stesso. Il parametro strErrorMessage contiene il messaggio di errore, il parametro ErrorNumber il numero di riferimento dell'errore.

| **Numero Errore** | **Costante enumerata** |
|---|---|
| 1 | ComPortGenericError |
| 10 | ComPortOlreadyOpen |
| 11 | ComPortNotAvailable |
| 1001 | ComPortEventBreak |
| 1004 | ComPortEventFrame |
| 1006 | ComPortEventOverrun |
| 1008 | ComPortEventRxOver |
| 1009 | ComPortEventRxParity |
| 1010 | ComPortEventTxFull |
| 1011 | ComPortEventDCB |
| 10000 | GenericError |
| 10001 | StringNotValid |

## *Costanti e loro valori:*

enmBoxStatus
BoxClosed = 0
BoxOpen = 1

enmControlBallStatus
Disconnected = 0
Connected = 1

BoxArray
Box1 – Box96 (Byte)

ControlBallArray
ControlBall1 – ControlBall8 (Byte)