



[Home Page](#) 
[Informazioni](#) 
[Aiuto](#) 


Riprodurre un file MIDI presente in un file di risorse

http://www.vbsimple.net/activity/act_16.htm

Richiesta di: [Claudio Gucchierato](#) - 7 Giugno 2001

Difficoltà:  3 / 5

Ho la necessità di trovare un codice funzionante per riprodurre dei files MIDI inseriti in un file di risorse .RES.





Il problema in questione è alquanto semplice. In questo progetto sarà utilizzato un [file di risorse](#)  contenente 4 files audio MIDI. Per comprendere come creare un file di risorse consultare l'[HowTo apposito](#) e la pagina dedicata alle [Informazioni aggiuntive](#).

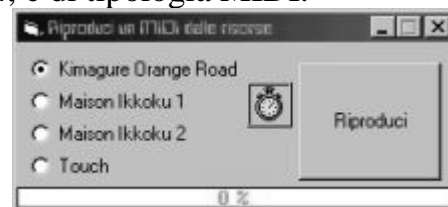
Il nostro file di risorse da compilare sarà il seguente:

```
#define MIDI1 1001
#define MIDI2 1002
#define MIDI3 1003
#define MIDI4 1004

MIDI1 MIDI KOR.MID
MIDI2 MIDI MIK1.MID
MIDI3 MIDI MIK2.MID
MIDI4 MIDI TOUCH.MID
```

Le quattro risorse saranno quindi numerate da 1001 a 1004, e di tipologia MIDI.

Il nostro progetto si compone di un solo form contenente una [matrice](#) di 4 **OptionButton**  di nome **MusicaScelta** con indici dallo 0 al 3, un controllo **Timer**  di nome **Tempo** con [proprietà](#)  **Interval** impostata a 200, un **CommandButton**  di nome **Riproduci** ed un controllo [FBIShapeProgressBar](#) di nome **Avanzamento**.




Nel momento in cui l'utente premerà il pulsante **Riproduci**, sarà verificato quale degli **OptionButton** è selezionato ed in base alla scelta, sarà estratta la risorsa relativa. La risorsa sarà salvata in un file temporaneo di nome **MIDIRIS.MID** posto nella cartella **TEMP** del sistema. Salvato il file, verrà lanciata la sua esecuzione attraverso le [MCI](#) (*Media Control Interface* - Interfaccia di controllo dei media).

1. Option Explicit
2. Private Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA" (ByVal lpstrCommand As String, ByVal lpstrReturnString As String, ByVal uReturnLength As Long, ByVal hwndCallback As Long) As Long
- 3.

Tutte le chiamate MCI saranno fatte mediante una sola funzione [API](#): **mciSendString**. Essa riceve una stringa di comandi (*lpstrCommand*) e la interpreta per effettuare le chiamate all'MCI; accetta anche un parametro stringa (*lpstrReturnString*) per contenere le risposte

alle interrogazioni MCI; il terzo parametro è l'ampiezza del [buffer](#) utilizzabile per le risposte e l'ultimo parametro consente di passare un [puntatore](#) ad una funzione [Callback](#) per processare i [messaggi](#) provenienti dall'MCI.

Il codice principale sarà posto nella routine che gestisce l'[evento](#)  Click sul pulsante **Riproduci**.

```
4. Private Sub Riproduci_Click()  
5.     Dim FILENR As Integer  
6.     Dim TEMPFILE As String  
7.     Dim BUFFERDATI() As Byte  
8.  
9.     Select Case True  
10.        Case MusicaScelta(0).Value: FILENR = 1001  
11.        Case MusicaScelta(1).Value: FILENR = 1002  
12.        Case MusicaScelta(2).Value: FILENR = 1003  
13.        Case MusicaScelta(3).Value: FILENR = 1004  
14.        Case Else: Exit Sub  
15.     End Select  
16.     BUFFERDATI = LoadResData(FILENR, "MIDI")
```

Alla riga 7 viene dichiarata una matrice di bytes di nome **BUFFERDATI**. Essa sarà utilizzata per leggere il contenuto della risorsa e scriverlo nel file temporaneo.

Alle righe 9-15 viene verificato quale dei 4 `OptionButton` è selezionato ed in base alla scelta assegna un valore alla variabile **FILENR**. In realtà la riga 14 non sarebbe necessaria ma abbiamo voluto aggiungerla per dare una certa pulizia e leggibilità di codice.

Alla riga 16 viene infine letta la risorsa il cui numero è contenuto in **FILENR**, di tipo *MIDI*. Il suo contenuto è salvato nella matrice **BUFFERDATI**.

```
17.     FILENR = FreeFile  
18.     TEMPFILE = Environ("TEMP")  
19.     If Right(TEMPFILE, "1") <> "\" Then TEMPFILE = TEMPFILE & "\"  
20.     TEMPFILE = TEMPFILE & "MIDIRIS.MID"  
21.     Call mciSendString("STOP MIDIRIS", ByVal 0&, ByVal 0&, ByVal 0&)  
22.     Call mciSendString("CLOSE MIDIRIS", ByVal 0&, ByVal 0&, ByVal 0&)  
23.     Open TEMPFILE For Output As FILENR  
24.     Close FILENR  
25.     Open TEMPFILE For Binary As FILENR  
26.     Put #FILENR, , BUFFERDATI  
27.     Close FILENR
```

Alla riga 17 viene recuperato l'[handle](#) del primo file libero e salvato in **FILENR**. Subito in seguito viene recuperata la posizione della cartella *TEMP* del sistema mediante lettura della variabile d'ambiente **TEMP**. Se il nome di tale cartella non termina con un \ esso sarà aggiunto (riga 19). Questo semplicissimo controllo evita problemi quando la cartella *TEMP* è in realtà la radice del disco.

Il nome del file temporaneo viene completato aggiungendo il nome *MIDIRIS.MID*.

Alle righe 21 e 22 viene fatta la prima chiamata alla funzione *mciSendString*. Tali due righe interrompono l'esecuzione del brano precedente di nome *MIDIRIS* e chiudono l'handle MCI aperto. Quest'operazione è fondamentale se l'utente sta ascoltando un file mediante questo programma e decide di cambiare brano e ne richiede un altro. Il brano precedente sarà prima arrestato e poi chiuso per permettere la cancellazione del file.

Alle righe 23 e 24 abbiamo un'operazione buffa: viene aperto il file temporaneo in

modalità output e subito in seguito viene chiuso. Questo fa sì che se il file esiste venga azzerato automaticamente. L'operazione è un'alternativa al controllo dell'esistenza e della successiva cancellazione del file temporaneo.

Alla righe 25-27 viene finalmente aperto il file temporaneo in modalità binaria ed il contenuto della matrice **BUFFERDATI** viene salvato in esso. Il file viene subito dopo chiuso.

```

28.      Call mciSendString("OPEN " & TEMPFILE & " TYPE SEQUENCER ALIAS MIDIRIS", ByVal
    0&, ByVal 0&, ByVal 0&)
29.      Call mciSendString("PLAY MIDIRIS FROM 0", ByVal 0&, ByVal 0&, ByVal 0&)
30.      TEMPFILE = Space(256)
31.      Call mciSendString("STATUS MIDIRIS LENGTH", TEMPFILE, 255, ByVal 0&)
32.      Avanzamento.Min = 0
33.      Avanzamento.Value = 0
34.      Avanzamento.Max = CLng(TEMPFILE)
35.      Tempo.Enabled = True
36. End Sub
37.
```

Dopo questi preliminari sarà possibile lanciare la riproduzione del file temporaneo salvato. Alla riga 28 viene aperto il file temporaneo mediante l'istruzione **MCI OPEN**. Essa richiede il nome del file da aprire ed il nome simbolico da assegnargli (**ALIAS MIDIRIS**). Opzionalmente sarà possibile (e consigliabile) fornire il tipo di lettore da utilizzare, nel nostro caso il sequencer MIDI (**TYPE SEQUENCER**).

Aperto il file sarà possibile lanciare la riproduzione mediante comando **MCI PLAY**. Esso richiede il nome simbolico utilizzato per aprire il file ed il punto di inizio della riproduzione (**FROM 0**).

A questo punto il file MIDI è in esecuzione. Per regolare l'andamento del nostro file MIDI leggeremo la lunghezza del file MIDI mediante l'istruzione **MCI STATUS**. Essa richiede il nome simbolico (**MIDIRIS**) seguita dal parametro-operazione **LENGTH**. Il risultato della richiesta sarà salvato nella stringa **TEMPFILE**.

Ottenuta la durata del brano viene inizializzata la barra di avanzamento impostandone valori minimo, attuale e massimo. Solo in seguito viene avviato il Timer che scandisce l'avanzamento del brano (riga 35).

```

38. Private Sub Form_Unload(Cancel As Integer)
39.      Call mciSendString("STOP MIDIRIS", ByVal 0&, ByVal 0&, ByVal 0&)
40.      Call mciSendString("CLOSE MIDIRIS", ByVal 0&, ByVal 0&, ByVal 0&)
41. End Sub
42.
```

Nel momento in cui l'utente chiude il programma è buona regola interrompere anche la musica e chiudere il file aperto (righe 39 e 40).

```

43. Private Sub Tempo_Timer()
44.      Dim BUFFER As String
45.      BUFFER = Space(256)
46.      Call mciSendString("STATUS MIDIRIS POSITION", BUFFER, 255, ByVal 0&)
47.      Avanzamento.Value = Val(BUFFER)
48. End Sub
49.
```

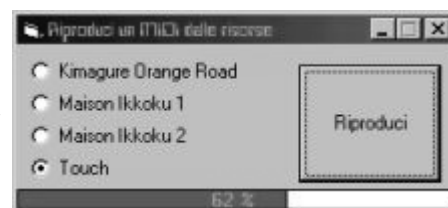
Il Timer Tempo scandirà l'avanzamento del brano ogni 200 millisecondi. Pertanto ogni

volta che scatta l'evento Timer sarà necessario leggere il punto di avanzamento del brano MIDI mediante l'istruzione MCI **STATUS** con il parametro **POSITION**. Il risultato della richiesta viene salvato nella variabile **BUFFER** e solo in seguito viene aggiornata la posizione corrente di Avanzamento con il nuovo valore recuperato.

```
50. Private Sub Avanzamento_Massimo()  
51.     Tempo.Enabled = False  
52. End Sub
```

Questa piccola quanto sciocca routine fa sì che il Timer venga arrestato al completamento della riproduzione del brano, cioè quando il valore corrente della barra di avanzamento raggiunge il massimo consentito ovvero la completa durata del brano. Ciò serve soltanto ad evitare qualche spiacevole ed inutile lampeggiamento grafico dovuto al continuo aggiornamento video.

Il programma è molto semplice ed istintivo all'uso. Basterà all'utente scegliere un brano tra i quattro a disposizione, premere il pulsante Riproduci ed ascoltare le note del brano.



Man mano che il brano procede la barra di avanzamento procede il suo corso.

L'utilizzo delle funzioni MCI semplifica enormemente il lavoro al programmatore. Inoltre esse non richiedono alcun controllo aggiuntivo che rischia di creare problemi durante l'installazione di un programma del genere.

[Fibia FBI](#)
25 Giugno 2001



[Torna all'introduzione delle Richieste dei lettori](#)
