ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Depto. de Engenharia Mecânica

Trabalho de Graduação

CONTROLADOR DE TIMBRES PARA TECLADOS MUSICAIS

Pedro Balbachevsky
Orientador: Prof. Lucas Moscatto

SÃO PAULO
1997

Dedico este trabalho à minha mãe, que sempre me mostrou o caminho certo a ser seguido através de seu exemplo de vida.

AGRADECIMENTOS

Agradeço a todos os amigos, colegas de curso, professores e funcionários que contribuíram direta ou indiretamente para a realização deste trabalho, os quais deixo de nomear para que nenhum seja esquecido ou aparente menor importância.

SUMÁRIO

1. Introdução

1.1 O problema

Nas últimas décadas, o ramo do entretenimento sofreu um impulso vertiginoso devido à sua capacidade de assimilar e combinar tecnologias novas. É incansável a busca pela melhoria da qualidade de dispositivos com características cada vez mais atraentes ao público como o som e a imagem.

No setor específico de shows musicais, nos quais os investimentos são crescentes e atraem multidões cada vez maiores, há problemas desafiadores a serem resolvidos. Seja na iluminação, potência de som, efeitos de voz, etc... , as soluções são cada vez mais engenhosas.

O problema a ser resolvido por este projeto de formatura é bem específico: o controle ágil e simples da escolha e mudança de timbres para um teclado musical.

O tecladista normalmente está submetido à condições adversas durante uma *performance* ao vivo:

- iluminação reduzida, inexistente ou desorientadora;
- volume de som no palco muito alto;
- condições emocionais extremas;
- necessidade de controle rápido e preciso de timbres.

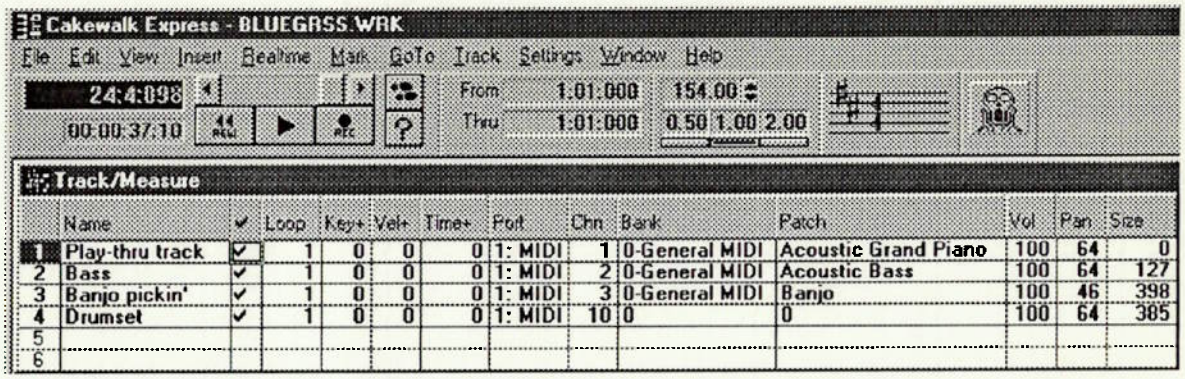
1.2 Algumas soluções existentes

A mudança convencional de timbres requer o uso de, normalmente, três a cinco botões diferentes e, além de requerer memorização, ocupa as mãos do tecladista durante preciosos segundos. Algumas soluções criadas para resolver este problema estão descritas nos itens seguintes.

1.2.1 Tracks de sequencers

Uma solução muito popular entre tecladistas é o uso de "tracks" (trilhas) do sequencer como armazenadores de timbres.

Tracks ou *trilhas* são usualmente usados para gravação em separado de canais diferentes existentes numa música. No entanto, escolhendo o timbre conveniente para cada trilha (mesmo que estejam vazias), pode-se trocar de timbre usando apenas um botão: aquele responsável pelo incremento/decremento do número da trilha corrente.



	Name	✔	Loop	Key+	Vel+	Time+	Port	Chn	Bank	Patch	Vol	Pan	Size
1	Play-thru track	✔	1	0	0	0	1: MIDI	1	0-General MIDI	Acoustic Grand Piano	100	64	0
2	Bass	✔	1	0	0	0	1: MIDI	2	0-General MIDI	Acoustic Bass	100	64	127
3	Banjo pickin'	✔	1	0	0	0	1: MIDI	3	0-General MIDI	Banjo	100	46	398
4	Drumset	✔	1	0	0	0	1: MIDI	10	0	0	100	64	385
5													
6													

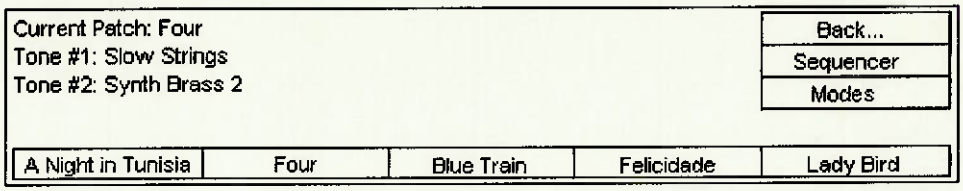
Esta é uma tela do software *Cakewalk Express*, muito popular entre os músicos que trabalham com o sequenciamento de músicas. À esquerda da tela estão os nomes das trilhas e pode-se perceber que a trilha corrente é a de número 1. Com uma única tecla (no caso, a seta para baixo), pode-se mudar de trilha e assim mudar o timbre de saída do sequencer conectado ao computador.

Apesar de parecer uma solução razoável, envolve o conhecimento prévio da sequência de músicas a ser tocada. Além disso, na maioria dos sequencers, o processo de leitura dos arquivos do disco para a memória pode ser demorado.

1.2.2 Patches

"Patches" são funções especiais presentes nos teclados mais novos do mercado capazes de armazenar configurações estáticas de timbres, isto é, escolhas fixas de timbres para diferentes músicas.

Quando o usuário escolhe uma das músicas da tela, o teclado é dividido, por exemplo, em duas regiões distintas com timbres diferentes. Outra possibilidade é a mudança de timbres a partir da velocidade de ataque das teclas.

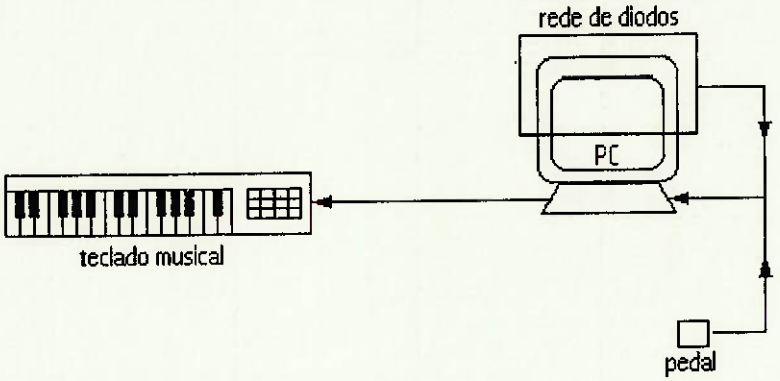


Os nomes das músicas são mnemônicos e isto elimina o problema de memorização de timbres. Porém, apenas dois timbres de cada vez podem ser escolhidos e uma terceira mudança se torna trabalhosa.

2. Apresentação do projeto

O sistema desenvolvido consiste em:

- Um computador PC conectado a um teclado musical através de interface MIDI (Musical Instrument Digital Interface);
- Um pedal liga-desliga conectado ao computador via porta paralela;
- Uma rede de diodos emissores/receptores infra-vermelhos conectados ao computador via porta paralela.

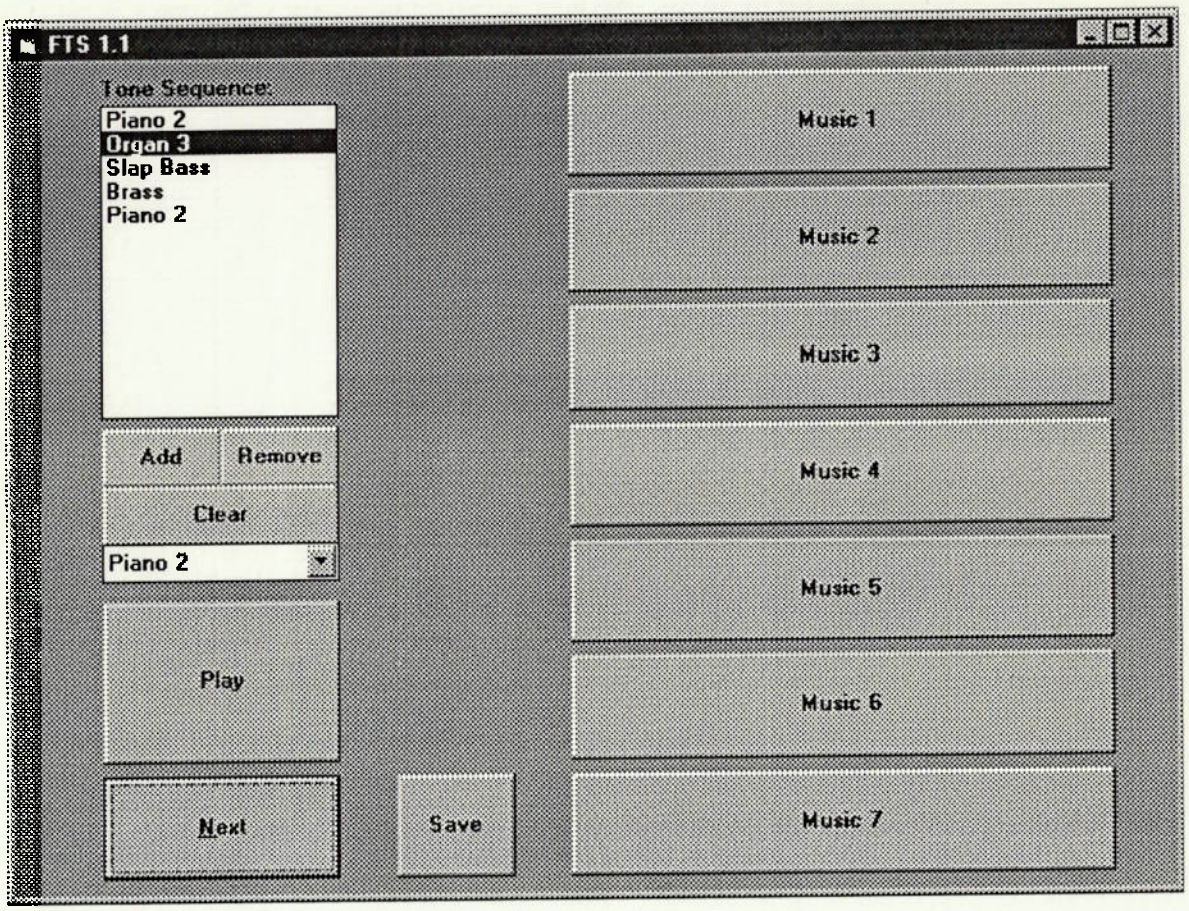


Através da tela do PC o usuário pode editar e armazenar sequências de timbres, bem como monitorar o timbre que está sendo usado no momento. Tudo isto através de um aplicativo *for Windows* específico.

O pedal provoca a troca de timbres ao longo da lista e permite ao usuário permanecer sempre com as duas mãos no teclado.

A rede de diodos permite a seleção de músicas através de "touch-screen", ou seja, o usuário toca diretamente na tela o botão correspondente à música desejada.

2.1 Interface com o usuário



2.1.1 Edição

A edição das várias sequências de timbres é feita através dos botões **Add**, **Remove**, **Clear** e através da "combo box" logo abaixo do botão **Clear**.

Primeiramente, com a lista de timbres vazia (caso não esteja vazia, basta utilizar o botão **Clear**), o usuário deve executar a seguinte sequência de operações:

1) Abrir a "combo box" citada acima;

2) Clicar no timbre escolhido;

3) Pressionar o botão **Add**;

4) Caso a lista não esteja terminada, voltar ao passo 1 repetindo as operações para o próximo timbre.

Caso cometa algum erro, basta clicar no timbre a ser removido e, em seguida, pressionar o botão **Remove**.

Após a edição da lista, basta pressionar o botão **Save** para que esta seja guardada no espaço da música atual.

Obs: Note que a música atual é aquela que teve seu botão correspondente pressionado pela última vez. Caso nenhum botão de música tenha sido pressionado, a música atual é a Música 1.

2.1.2 Operações *Real-Time*

No momento de execução de uma música, o usuário deve, inicialmente, entrar no modo *Real-Time*, pressionando o botão **Play**. A partir deste momento, serão usados apenas o pedal de controle, e a rede de diodos. O primeiro faz as vezes do botão **Next**, trocando de timbres ao longo da lista visível na tela e a segunda possibilita a seleção de músicas via touch-screen.

Obs: Quando o cursor chegar ao último timbre da lista, ele voltará ao primeiro, caso o pedal seja pressionado uma vez mais.

3. Implementação e Funcionamento

3.1 O programa

O programa para o aplicativo *for Windows* criado foi desenvolvido em linguagem *Visual Basic 3.0*. Por se tratar de uma linguagem direcionada à objetos, sua listagem é composta de vários sub-códigos, cada um referente a um objeto (botões, boxes, etc...). Estes códigos estão devidamente listados no *Apêndice A*.

3.1.1 Estrutura de dados

A estrutura de dados utilizada é a mais simples possível, composta de arquivos randômicos. Isto porque, devido ao tamanho reduzido dos arquivos, não há necessidade de atualizá-los de maneira sequencial ou através da criação de bancos de dados. A cada comando **Save**, o arquivo antigo é apagado totalmente e a lista corrente assume seu lugar.

Os códigos responsáveis pelo tratamento de arquivos estão também listados no *Apêndice A* (vide procedimento Le_Mus e botão **Save**).

3.1.2 Mensagens MIDI

3.1.2.1 O que é MIDI?

O padrão MIDI (Musical Instruments Digital Interface) surgiu em 1982 na convenção da NAMM (National Association of Music Manufacturers) quando duas das maiores empresas produtoras de instrumentos musicais eletrônicos (Korg e Kawai) tornaram possível uma comunicação digital entre seus produtos.

Através de um simples cabo serial de cinco pinos, mensagens assíncronas em blocos de 10 bits podem ser trocadas entre os dispositivos a uma velocidade de 31250 bits por segundo.

Este padrão tomou rapidamente conta do mercado e hoje qualquer dispositivo eletrônico, desde simples teclados controladores a mixers de estúdio com 64 canais inclui um conjunto de portas MIDI.

3.1.2.2 Mensagens utilizadas

O número e a classificação das mensagens MIDI é extensa e cresce juntamente com a sofisticação dos produtos disponíveis no mercado.

Neste projeto, apenas será usada a mensagem de troca de timbre (*Program Change*) que tem o seguinte formato:

(Byte de identificação) + (1 ou 2 bytes de dados)

Exemplos:

➔ &H9C + &H50 + &H7F

(Canal 13, Nota 80, Velocidade 127)

➔ &H8C + &H50 + &H00

(Canal 13, Nota 80, Velocidade 0)

➔ &HCC + &H68

(Tone 104)

O uso extensivo destas mensagens pode ser verificado na listagem do código referente ao botão **Next** (Apêndice A).

3.1.3 Funções API

O sistema de multimídia do Windows fornece interface para várias atividades como:

- reprodução e gravação de áudio WAVE;
- sintetizador de áudio e MIDI;
- reprodução de animação;
- reprodução de vídeo;
- serviços para joysticks;
- etc...

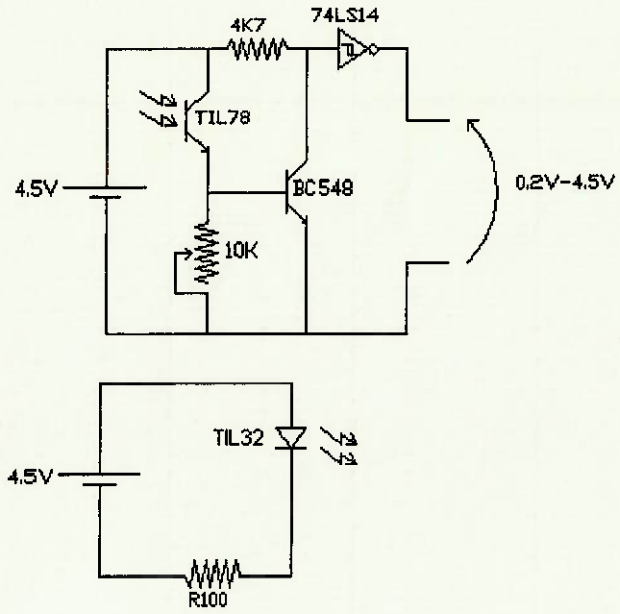
A maioria das funções responsáveis por estes serviços estão localizadas na DLL (*dinamic link library*) MMSYSTEM.DLL e podem ser utilizadas pelo *Visual Basic* desde que devidamente declaradas no programa. São as funções API (*Applications Programming Interface*).

Apenas cinco funções API são utilizadas neste projeto:

midiOutOpen:	abre o dispositivo externo (no caso, um sequencer);
midiOutShortMsg:	manda mensagem pela porta MIDI através de um argumento hexadecimal;
midiOutClose:	fecha o dispositivo externo;
CreateFile:	prepara a porta paralela para ser lida;
ReadFile:	lê o byte corrente na porta paralela.

Estas funções estão declaradas no módulo FTS_2.BAS, listado no Apêndice B.

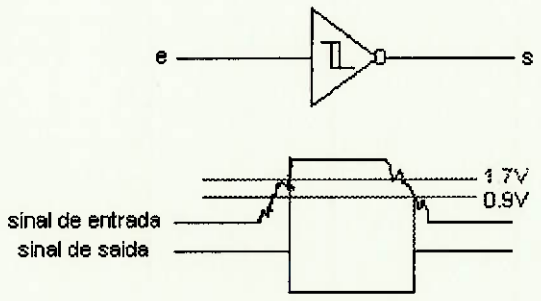
3.2 O hardware externo



O hardware externo, a ser acoplado na tela do computador portátil, engloba o pedal de controle e a rede de diodos num mesmo circuito. Cada par de componentes ópticos (LED e transistor) envolve o seguinte sub-circuito:

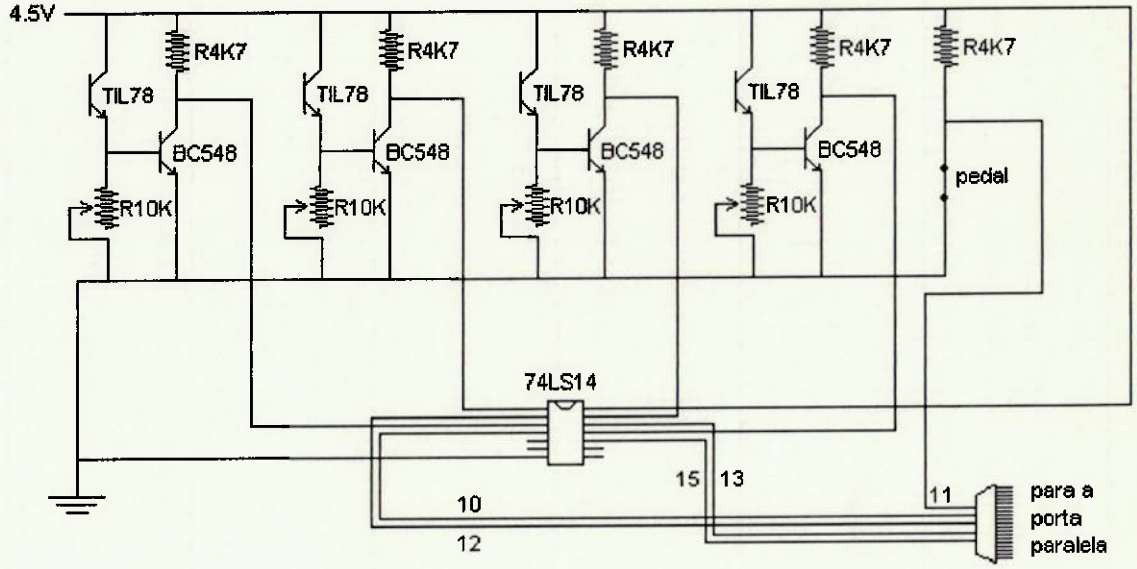
Este circuito possui algumas características importantes:

- é alimentado por três pilhas comuns AA de 1.5V, o que garante sua portabilidade;
- possui um *trimpot* de 10kΩ que regula a corrente no seu ramo e, com isso, a sensibilidade do transistor foto-receptor (TIL 78);
- possui um *Schmidt-Trigger* que elimina ruídos e inverte sinal de saída a seguinte forma:

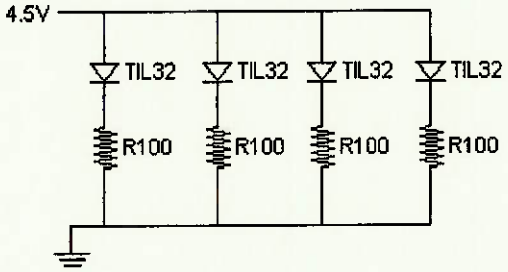


Este tratamento garante uma leitura mais estável na porta paralela do computador.

O circuito completo, com quatro conjuntos ópticos e mais o pedal de controle (chave normalmente fechada), é o seguinte:



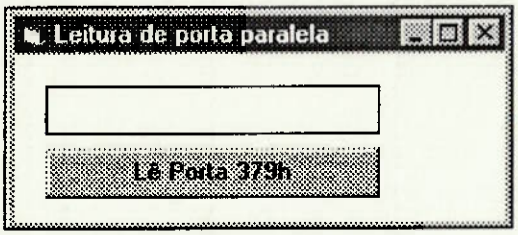
Este circuito foi implementado numa placa de montagem à esquerda do monitor de vídeo do computador. À direita do mesmo, foi implementada uma segunda placa, menor, com os LED's emissores:



4. Testes

4.1 Testes de software

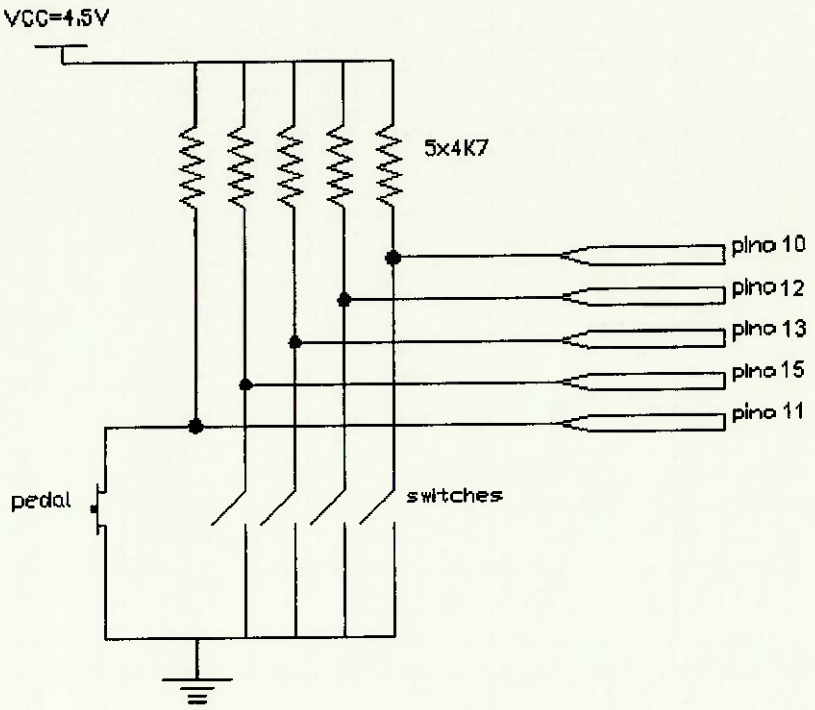
As funções API utilizadas garantem o bom funcionamento das subrotinas mais importantes do software: o envio de mensagens MIDI e a leitura da porta paralela. Para testar tais subrotinas, foram criados os seguintes programas:



As telas são auto-explicativas e os programas utilizam as mesmas subrotinas que o programa principal.

4.2 Testes de hardware

Para testar a saída do sinal de cada par óptico, foi o utilizado o circuito individual mencionado no item anterior e para testar a recepção de bytes na porta paralela, foi implementado o seguinte circuito:



APÊNDICE A - Objetos utilizados e seus respectivos códigos

Declarações globais:

```
'Mus é o número da música atual
Dim Mus As Integer

'hMidiOut é parâmetro de função API
Dim hMidiOut As Integer

'MidiDeviceOpen é usada como status booleano
Dim MidiDeviceOpen As Integer
```

Botão Remove:

```
Sub Remove_Click ()

'Remove item da lista de timbres

If List1.ListIndex <> -1 Then
        List1.RemoveItem List1.ListIndex
End If

End Sub
```

Botão Add:

```
Sub Add_Click ()

'Adiciona item à lista

List1.AddItem Combo1.Text
End Sub
```

Botão Clear:

```
Sub Clear_Click ()

'Limpa lista

List1.Clear
End Sub
```

Abertura do Formulário:

```
Sub Form_Load ()

'Procedimento realizado qdo o usuário inicia o programa

Reset
Mus = 1
```

```
'Lista de timbres disponível para escolha

Combo1.AddItem "Piano 2"
Combo1.AddItem "Vibraphone"
Combo1.AddItem "Marimba"
Combo1.AddItem "Organ 3"
Combo1.AddItem "Bandoneon"
Combo1.AddItem "Overdriven Gt."
Combo1.AddItem "Slap Bass"
Combo1.AddItem "Slow Strings"
Combo1.AddItem "Brass"
Combo1.AddItem "Flute"
Combo1.Text = Combo1.List(0)

'Abertura do dispositivo MIDI

MidiDeviceOpen = False
RetValue = midiOutOpen(hMidiOut, MIDI_MAPPER, 0, 0, 0)
MidiDeviceOpen = True

End Sub
```

Fechamento do Formulário:

```
Sub Form_Unload (Cancel As Integer)

'Fecha dispositivo MIDI

RetValue = midiOutClose(hMidiOut)
End Sub
```

Botão Música 1:

```
Sub Mus1_Click ()

'Atualiza música atual e chama procedimento Le_Mus
'Obs: Todos os botoes Música# sao semelhantes

Mus = 1
Le_Mus (1)
End Sub
```

Botão Next:

```
Sub Next_Click ()

'Incrementa posicao do cursor da lista
If List1.ListIndex <> List1.ListCount - 1 Then
        List1.ListIndex = List1.ListIndex + 1
Else
        List1.ListIndex = 0
End If
```

```
'Reconhece item

Select Case List1.List(List1.ListIndex)

Case "Piano 2"
   MidiShortMessage = &H1C0

Case "Vibraphone"
   MidiShortMessage = &HBC0

Case "Marimba"
   MidiShortMessage = &HCC0

Case "Organ 3"
   MidiShortMessage = &H12C0

Case "Bandoneon"
   MidiShortMessage = &H17C0

Case "Overdriven Gt."
   MidiShortMessage = &H1DC0

Case "Slap Bass"
   MidiShortMessage = &H24C0

Case "Slow Strings"
   MidiShortMessage = &H31C0

Case "Brass"
   MidiShortMessage = &H3DC0
```

```
Case "Flute"
   MidiShortMessage = &H49C0

End Select

'Manda mensagem MIDI adequada

RetValue = midiOutShortMsg(hMidiOut, MidiShortMessage)

End Sub
```

Procedimento Le_Mus:

```
Sub Le_Mus (n)

'Transfere sequencia de timbres do arquivo para a lista

List1.Clear
Open "MUSICA" & n & ".DAT" For Random As n
While Not EOF(n)
   Get n, , Tone$
   List1.AddItem Tone$
Wend
Close Mus

'Remove linha em branco que vem junto do arquivo

List1.RemoveItem List1.ListCount - 1
End Sub
```

Botão Salva:

```
Sub Salva_Click ()

'Salva sequencia de timbres

Close Mus
Kill "MUSICA" & Mus & ".DAT"
Open "MUSICA" & Mus & ".DAT" For Random As Mus

For i = 0 To List1.ListCount - 1
   Tone$ = List1.List(i)
   Put Mus, , Tone$
Next

Close Mus

End Sub
```

APÊNDICE B - Módulo FTS_2.BAS (funções API)

'Declaração das funçoes API utilizadas

Declare Function *midiOutOpen* Lib "MMSystem" (hMidiOut As Integer, ByVal DeviceId As Integer, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Integer

Declare Function *midiOutShortMsg* Lib "MMSystem" (ByVal hMidiOut As Integer, ByVal MidiMessage As Long) As Integer

Declare Function *midiOutClose* Lib "MMSystem" (ByVal hMidiOut As Integer) As Integer

Declare Function *CreateFile* Lib "kernel32" (ByVal lpFileName As String, ByVal dwDesiredAccess As Long, ByVal dwShareMode As Long, lpSecurityAttributes As Long, ByVal dwCreationDistribution As Long, dwFlagsAndAttributes As Long, ByVal hTemplateFile As Integer) As Integer

Declare Function *ReadFile* Lib "kernel32" (ByVal hFile As Integer, ByVal lpBuffer As Long, ByVal nNumberOfBytesToRead As Long, ByVal lpNumberOfBytesToRead As Integer, ByVal lpOverlapped As Integer) As Integer

Global Const MIDI_MAPPER = -1

REFERÊNCIAS BIBLIOGRÁFICAS

1. JAROL, Scott. **Visual Basic para Multimídia**. Rio de Janeiro: Campus, 1995.

2. JENNINGS, Roger. **Windows 3.1 Multimídia - ferramentas poderosas**. Rio de Janeiro: Berkeley, 1993.

3. FRANKLIN, Carl. **Ask the VB pro**. http://www.inquiry.com/thevbpro/

4. Manuais e *Data Sheets* de TTL's.

5. Manuais e *Data Sheets* de Hardware e Interfaceamento.