

Lok mit Videokamera

In letzter Zeit werden bei EBAY Funk-Video-Kameras für wenig Geld angeboten, die sich zum Einbau in Modellfahrzeuge eignen. Damit kann der Modelleisenbahner seine Anlage aus Lokführersicht erleben.

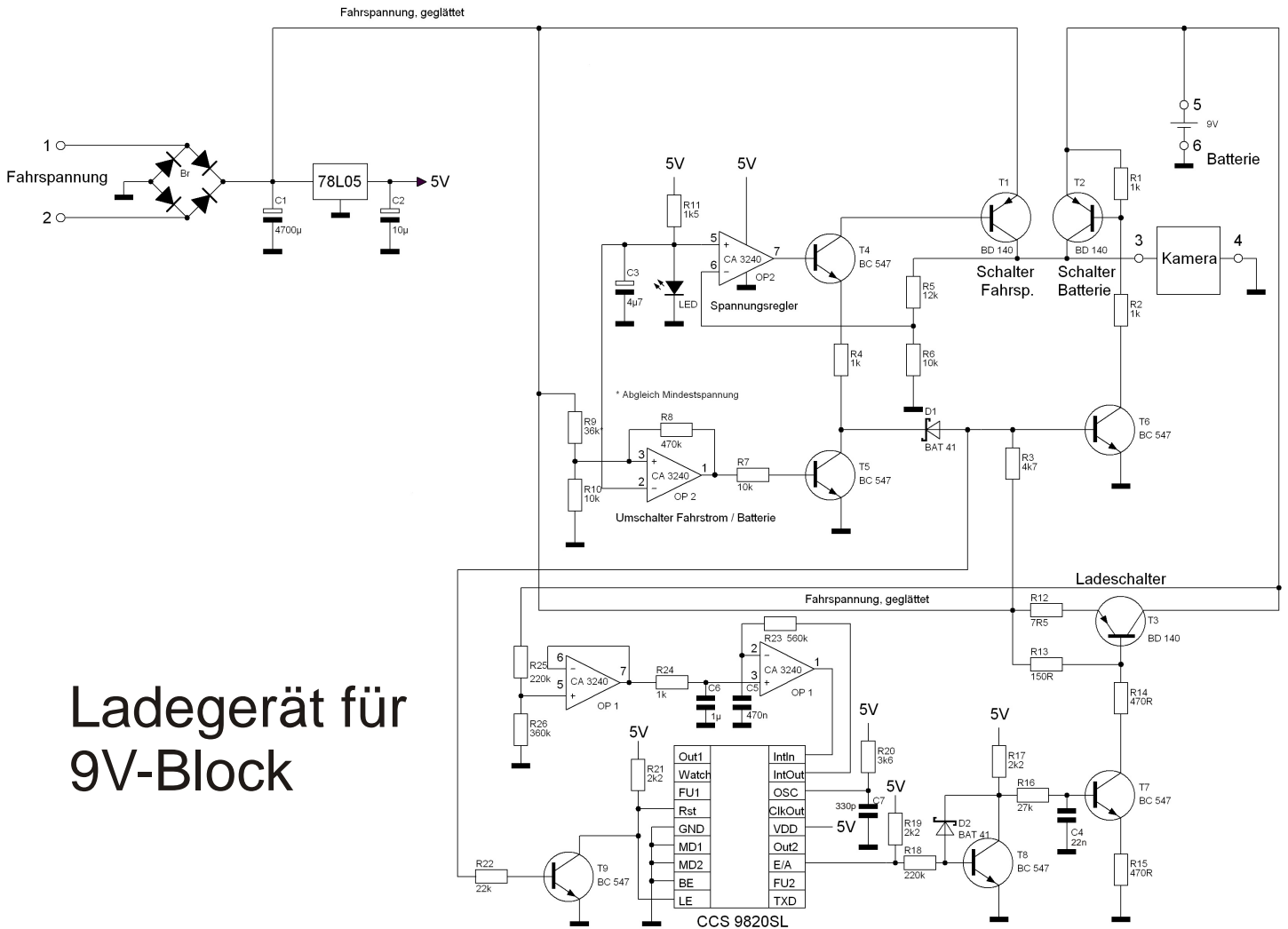
Die Videokamera mit dem Sender wird in die Lok eingebaut, der Empfänger wird an eine TV-Karte angeschlossen, die als Computerzubehör zu haben ist. Man kann aber auch einen Fernseher benutzen.

In **Visual Basic** kann ein Videofenster programmiert werden, das als Lokführerstand gestaltet werden kann.

Die Funk-Kamera wird mit einem 9V-Block betrieben, für den man zweckmäßigerweise einen Akku benutzen sollte.

Die folgende Schaltung sorgt nun dafür, daß dieser Akku im Fahrbetrieb automatisch nachgeladen werden.

Den Ladevorgang regelt ein CCS 9620SL in Verbindung mit OP1 über den Transistor T3. Solange genügend Fahrspannung vorhanden ist, wird die Kamera aus dieser über T1 versorgt. Anderenfalls bezieht sie den Strom über T2 aus der Batterie. Damit sich die Batterie bei längeren Betriebspausen nicht entlädt, sollte sie dann abgeschaltet werden. Die Schaltung eignet sich besonders gut für Impulsbreitenbetrieb, weil hier die volle Fahrspannung auch bei Langsamfahrt verfügbar ist.



Ladegerät für
9V-Block

Geräuschkulisse

Probeweise habe ich mit der Lochkreissäge einen Ausschnitt unter der Kirche mit der Hochzeitsszene in die Platte geschnitten und einen Lautsprecher (mit Gehäuse) eingebaut. Dabei zeigte sich, daß man den Ort von Klangereignissen sehr wohl orten kann.

Mit Hilfe der API-Funktionen kann man in **Visual Basic** MP3-Dateien abspielen, auch mehrere gleichzeitig. Auf einer speziellen CMS-Relaiskarte habe ich einen NF-Verstärker untergebracht, mit den Relais kann man einen oder mehrere Lautsprecher zuschalten.

Nun kann ich aus der Kirche heraus Glockengeläut oder auch Orgelmusik ertönen lassen. Unter dem Rummelplatz nebenan ist der nächste Lautsprecher untergebracht. Hier kann ich in einer Schleife vier Orchestrierteile abspielen. Der nächste Lautsprecher ist unter den Bahnsteigen eingebaut. Hier kann ich nun diverse Lokgeräusche abspielen.

Aber hier kommt noch eine zusätzliche Möglichkeit ins Spiel - der Sprachsynthesizer. Mit diesem VB-Zusatzsteuerelement kann man Texte als Sprache ausgeben. Diese Texte können als Strings vorgefertigt abgespeichert und beliebig zusammengesetzt werden. Die manchmal exotisch anmutende Sprechweise kann man durch bewußt falsche Schreibweise und Sonderzeichen korrigieren. Bessere Ergebnisse erhält man mit der Software **Logox**. Sie ist für die deutsche Sprache

vorgesehen und hat allerlei Optionen, von Hall bis Dialekt. Man kann dazu auch eine SDK (Entwickler-Werkzeug, in verschiedenen Programmiersprachen nutzbar) erhalten, aber der einfachere und den Computer weniger belastende Weg ist, die Soundtracks als WAV oder MP3 zu produzieren.

So kann man automatische Bahnhofsdurchsagen programmieren, die zusätzlich mit Klangereignissen, z.B. einem Gong, garniert werden können.

Allerdings gibt es mit den **API-Funktionen** ein Problem: Man kann zwar mehrere Soundereignisse gleichzeitig abspielen, aber, da sie aus derselben Soundkarte stammen, nicht einzeln auf die Lautsprecher verteilen.

Abhilfe kann man schaffen, wenn man die Stereo-Kanäle mehrerer Soundkarten gleichzeitig einzeln steuert. Das ist mit der **DirectX**-SDK oder dem Freeware-Soundsystem **FMOD** machbar. FMOD ist insofern besser, weil hier auch MP3-Dateien benutzt werden können. Allerdings stürzt mein alter Win98-Rechner dabei gern ab, wenn zuviel Sound gleichzeitig läuft.

Es müßte eigentlich auch machbar sein, die 3-D-Sound-Kanäle einzeln zu nutzen, aber leider habe ich noch keine verständliche Dokumentation mit funktionierenden Beispielen gefunden. Hier nehme ich gern jeden Rat dankbar an.

Programmierung mit API-Funktionen

Zunächst braucht man einige API-Deklarationen, die man der Übersicht halber in einem Modul "Sound" unterbringt.

```
Public Declare Function mciSendString Lib "winmm.dll" _
    Alias "mciSendStringA" (ByVal lpszCommand As String, _
    ByVal lpszReturnString As String, _
    ByVal cchReturnLength As Long, _
    ByVal hwndCallback As Long) As Long
Private Declare Function GetShortPathName Lib "kernel32" _
    Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _
    ByVal lpszShortPath As String, _
    ByVal cchBuffer As Long) As Long
```

'Zum Abspielen der MP3-Dateien braucht man die Funktionen **MP3_play** und **MP3_Stop**

```
Public Function MP3_Play(ByVal sFile As String, _
    ByVal sAlias As String) As Boolean
    Dim bResult As Boolean
    Dim sBuffer As String
    Dim IResult As Long
    sBuffer = Space$(255)
    IResult = GetShortPathName(sFile, sBuffer, Len(sBuffer))
    If IResult <> 0 Then
        sFile = Left$(sBuffer, InStr(sBuffer, vbNullChar) - 1)
        'MCI öffnen
        IResult = mciSendString("open " & sFile & _
            " type MPEGVideo alias " & sAlias, 0, 0, 0)
        If IResult = 0 Then
            If mciSendString("play " & sAlias & _
                " from 0", 0, 0, 0) = 0 Then
                bResult = True
            End If
        End If
    End If
    MP3_Play = bResult
End Function
```

```
Public Sub MP3_Stop(ByVal sAlias As String)
    mciSendString "stop " & sAlias, 0, 0, 0
    mciSendString "close " & sAlias, 0, 0, 0
End Sub
```

```
Public Function SendCommand(ByVal sCmd As String) As Long
    Dim IResult As Long
    Dim sReturn As String
    sReturn = Space$(256)
    IResult = mciSendString(sCmd, sReturn, Len(sReturn), 0&)
    SendCommand = Val(sReturn)
End Function
```

Wenn man eine MP3 mit einer Taste starten und beim nächsten Klick stoppen will, schreibt man:

```
Private Sub TasteMP3_Click()
    If Tastenkennung = False Then
        MP3 = "Pfad"
        MP3_Play MP3, "Alias"
    Else
        MP3_Stop "Alias"
    End If
    Tastenkennung = Not Tastenkennung
End Sub
```

Erst beim zweiten Klick wird das Abspielen beendet. Damit man das Abspielen automatisch steuern kann, braucht man Länge und aktuelle Position der MP3. Die dazu nötige ständige Abfrage organisiert man über einen Timer:

```
Private Sub TimerAktuell_Timer()
    TimerAktuell.Enabled = False
    If Tastenkennung = True Then
        PositionMP3 = SendCommand("status Alias position")
        LängeMP3 = SendCommand("status Alias length")
        If PositionMP3 = LängeMP3 Then MP3_Stop "Alias"
    End If
    ...
    TimerAktuell.Enabled = True
End Sub
```

Sprachausgabe

Dazu braucht man das Zusatzsteuerelement **DirectSS**, das kostenlos aus dem Internet zu haben ist. Man kann per Code wählen:

```
Name.CurrentMode = 1 'Frauenstimme  
Name.CurrentMode = 2 'Männerstimme
```

Die Lippengrafik kann man ausschalten:

```
Name.Visible = False
```

Der Sprechbefehl:

```
Name.Speak Tex$ & Zahl% & "Text"
```

Dabei werden Zahlen mit ausgesprochen. Leider ist das Steuerelement nicht dokumentiert. Experimentieren Sie einfach mit der Tastatur, um die Aussprache zu korrigieren.

Das Ende eines Sprechbefehls kann mit der Eigenschaft **Speaking** ermittelt werden. Das ist wichtig, um den Lautsprecher wieder ausschalten zu können oder hinterher noch eine MP3 abzuspielen. Die Variable ist **1**, solange ein Text gesprochen wird, anderenfalls **0**.

Beispiel zur Programmierung einer Durchsage-Automatik:

Per CMS kann man eine Fahrstraße stellen, indem man nacheinander auf die Tasten des Startgleises und

Open-Air-Konzert mit Lichtorgel

Als besonderen Gag habe ich eine Rockbühne auf der Anlage. Die Musik war weiter kein Problem, es kommt eine schön schlechte Live-Aufnahme als MP3-Datei zur Aufführung. Aber buntes Licht gehört auch dazu. Mit einem FFT-Filter kann man die Lautstärkewerte einzelner Frequenzen mithilfe eines VB-Programms ermitteln, auswerten und per COM-Port an einen Mikroprozessor, dieses Mal von Atmel, schicken. Der Mikrocontroller steuert mit den Lautstärke-Informationen drei superhelle

Soundausgabe per FMOD

Mit FMOD kann man jeden Stereo-Kanal zum Abspielen eines Mono-Streams benutzen. In dieser Version kann nur eine Soundkarte initiiert werden. Wenn mehrere benutzt werden sollen, müssen mehrere Instanzen von FMOD eingerichtet werden. Dieser Nachteil entfällt bei der neuen Version **FMODex**. Leider treten in den VB-Beispielen Fehler auf, die ich bisher mangels verständlicher Anleitung nicht klären konnte.

Mit dem folgendem Code wird ein Stream geladen:

Private Sub TasteStreamÖffnen_Click(index As Integer)

'Streams werden nicht komplett in den Speicher geladen

'Wenn ein Stream abgespielt wird, lädt FMOD nur den Teil, der gerade spielt in den Speicher

'Wird benutzt, wenn mit FMOD Musik gespielt werden soll

'Datei per Dialog wählen:

```
CommonDialog1.Filter = "Sound-Dateien (*.wav, *.mp3, *.ogg, *.wma)|*.wav;*.ogg;*.mp3;*.wma"
```

```
CommonDialog1.ShowOpen
```

'Fehlerbehandlung

```
If Not FileExist(CommonDialog1.filename) Then 'Funktion
```

```
    MsgBox "Datei existiert nicht oder keine Datei gewählt"
```

```
    Exit Sub
```

```
End If
```

'Stream öffnen:

```
stream(1) = FSOUND_Stream_Open(CommonDialog1.filename, FSOUND_MONO, 0, 0)
```

'Anzeige:

```
LabelStream(1) = CommonDialog1.filename
```

'Tasten steuern

des Zielgleises klickt. Die so vorgewählte Fahrstraße wird auf Ausführbarkeit geprüft (Verriegelung) und dann im Gleisbild angezeigt und die Stellvorgänge werden vorgenommen.

Zu den Stellvorgängen gehört dann auch die Bahnsteigdurchsage.

```
Kennung = true 'aufzurufendes Soundereignis
```

```
Gleis% = x 'zu nennendes Gleis
```

```
Bhf$ = "xxx" 'zu nennender Bahnhof
```

```
MP3 = "Pfad"
```

```
MP3_Play MP3, "Alias" 'Abspielen eines Gongs
```

Ein globaler Timer überprüft das Soundereignis.

```
If Kennung = True then
```

```
    PositionMP3 = SendCommand("status Alias position")
```

```
    LängeMP3 = SendCommand("status Alias length")
```

```
    If PositionMP3 = LängeMP3 Then
```

```
        MP3_Stop "Alias"
```

```
    If Durchsage = False then 'nur einmal sprechen
```

```
        Name.Speak "Text" & Bhf$ & "Text" & Zahl% & "Text"
```

```
        'in der Durchsage werden ein Bahnhof und
```

```
        ein Bahnsteig genannt
```

```
        If Name.Speaking = 0 Then Aus 'Durchsage fertig
```

```
        Durchsage = True
```

```
    End If
```

```
End If
```

Lumi-LEDs, die als Scheinwerfer fungieren.

Den FFT-Filter gibt es im FMOD-System, aber ich habe als erstes ein Beispiel mit dem FFTW-Filter in

http://actorics.de/rm_code/fft.htm

gefunden und bin, da es einwandfrei funktionierte, dabei geblieben.

Da der Code etwas umfangreicher ist, verzichte ich hier auf die Darstellung. Falls Sie sich dafür interessieren, können Sie per eMail nachfragen.

```

If stream(1) <> 0 Then 'erfolgreich geladen
  TasteStreamSchließen(1).Enabled = True
  TasteStreamÖffnen(1).Enabled = False
  TasteStreamPlay(1).Enabled = True
Else 'schiefgegangen
  MsgBox "Beim Öffnen des Streams ist ein Fehler aufgetreten!" & vbCrLf & _
    FSOOUND_GetErrorString(FSOOUND_GetError), vbOKOnly
End If
End Sub

```

'Per Taste abspielen:

```

Private Sub TasteStreamPlay_Click(index As Integer)
StreamChannel = FSOOUND_Stream_Play(1, stream(1))
Lautstärke(1) = FSOOUND_SetVolume(1, 255) 'Lautstärke
Kanal(1) = FSOOUND_SetPan(1, TextPan(1)) 'links-rechts
If StreamChannel <> 0 Then 'läuft
  TasteStreamPlay(1).Enabled = False 'Tastensteuerung
  TasteStreamStop(1).Enabled = True
  If Schleife = True Then 'mehrfach abspielen
    Antwort = FSOOUND_Stream_SetMode(stream(1), 2) 'FSOUND_LOOP_NORMAL
    Antwort = FSOOUND_Stream_SetLoopCount(stream(1), Int(TextLoop) - 1)
  Else
    Antwort = FSOOUND_Stream_SetMode(stream(1), 1) 'FSOUND_LOOP_OFF
  End If
Else 'schiefgegangen
  MsgBox "Fehler beim Abspielen des Streams!" & vbCrLf & _
    FSOOUND_GetErrorString(FSOOUND_GetError), vbOKOnly
End If
End Sub

```

'Stream stoppen:

```

Private Sub TasteStreamStop_Click(index As Integer)
FSOUND_Stream_Stop stream(1)
StreamChannel = 0
'Timer zur Kontrolle des Ablaufs
TimerAktuell.Enabled = False
'Tastensteuerung
TasteStreamPlay(index).Enabled = True
TasteStreamStop(index).Enabled = False
End Sub

```

Der Ablauf wird mit einem Timer kontrolliert:

```

Private Sub TimerAktuell_Timer()
TextLänge = FSOOUND_Stream_GetLength(stream(2)) 'Anzeige
TextPosition = FSOOUND_Stream_GetPosition(stream(2))
'Anzeige mit Progressbalken:
If Val(TextLänge) > 0 Then ProgressBalken.value = (TextPosition) * 100 / Val(TextLänge)
'Playliste abspielen:
If ProgressBalken.value = 100 Then
Aktuell = Aktuell + 1
If Aktuell > TextEinträge - 1 Then Aktuell = 0
  TextIndex = Aktuell
  ProgressBalken.value = 0
  PlayListe
End If
End Sub

```

Der Einfachheit halber wird die komplette **FMOD.BAS** in das Projekt geladen. Die Handhabung von **DirectX** ist ähnlich, nachteilig ist, daß Wav-Dateien benötigt werden.

Falls Sie die Sache interessiert, können Sie die komplette Beispiel-Datei per eMail bekommen.