Windows CE 3.0 Programmiertipp:
Sendersuchen und Datenbanken

Zunächst einmal: das, was hier abgebildet und beschrieben ist, läuft so nur mit der Kombination aus D2 und einem Siemenshandy. Für andere Kombinationen müsste das Programm und die Datenbank umgeschrieben werden. Da ich selbst keine anderen Handys und Netzkarten besitze, werde ich das Programm so beschreiben, dass ein anderer dies vielleicht für andere Kombinationen realisieren kann. In dem Artikel werden auch entscheidende Hilfe gegeben, wie man mit CE - Datenbanken umzugehen hat.

Dass man sich um die Sender von D1 oder D2 oder auch Viag und e+ interessiert, ist eigentlich nicht selbst verständlich. Man telefoniert mit dem Handy und wenn es klappt, so ist es in Ordnung. Wer sich jedoch für die technischen Zusammenhänge des GSM – Standards interessiert, der möchte vielleicht gerne wissen, wie es um sein Netzt bestellt ist. Nutzt man das Modem im Handy, so kann man es befragen, in welcher GSM-Zelle es gerade eingebucht ist. Es liefert einen Wert LAI (Location Aerea Information) und CI (Cell-Information).

Bei dem unten abgebildeten Beispiel wird gerade der CI-Wert 0CD1 vom Handy in Hexdarstellung geliefert. Wird dieser Hexwert dezimal umgerechnet, so erhält man den Wert 3281. Hieraus ergibt sich der Antennenname 328 und die Segmentnummer der Antenne 1. Die Location LAI ist die Umsetzung des Wertes 0122 in die Dezimalzahl 290. Der dritte Wert, der im Handy zu erkennen ist (679) ist der aufgerufene Datensatz aus der Datenbank, die die Beschreibung der Senderstandorte enthält. Schickt man den at-Befehl at+creg=?, dann liefert das Handy als Antwort die oben beschriebenen Werte.

Im Bild wurde eine Fahrt von Köln nach Düsseldorf gespeichert. An der Zahl 60 unter der Liste ist die Zahl der Antennen zu erkennen, in der man eingeloggt war. Immerhin 60 an der Zahl. Witzigerweise zeigte der Kilometerzähler am Ziel 60 Kilometer. Vielleicht kann man diese Methode ja als Tachoersatz nutzen.

Noch einmal das Prinzip: Das Handy wird im Sekundentakt nach den aktuellen CI-Werten abgefragt. Dieser Wert wird mit den Einträgen in der Liste verglichen. Ist er neu, so wird er eingetragen und die Datenbank mit dem CI-Wert durchsucht. Die Datenbank liefert dann den Standort der Antenne, wie hier Köln – Marienburg.....

Die abgespeicherten Dateien über die Standorte der Sendemasten sind recht umfangreich. So hat die Datei von NRW über 4400 verschiedene Antennennamen nur für das D2 - Netz. Will man diese

Dateien im PDA benutzen, so muss man sich überlegen, in welcher Form man auf die Daten zugreifen will. Nutzt man die Datei als normale Textdatei, so muss man alle Einträge durchsuchen, um die aktuellen Antennendaten zu erhalten. Ist die Antennen – ID relativ klein, so wird sie bei der sequentiellen Suchmethode recht schnell gefunden, da die Datei nach aufstrebenden Werten sortiert ist. Liegt sie jedoch relativ weit hinten, so sucht man bei CE entsprechend lange. Man könnte hier eine Randomdatei mit festen Satzlängen installieren. Besser ist es jedoch, wenn man eine Datenbank anlegt und mit SQL-Kommandos sucht.

Für viele, die am Anfang stehen, wird dies zuerst schrecklich klingen. Es ist so schwer jedoch nicht. Wer mit VB3/4/5/6 und Accessdatenbanken gearbeitet hat, der wird den Einstieg schneller finden. Hier jedoch kann man nicht einfach Controls mit Datenbankfunktionen belegen, wie dies in den VB-Versionen möglich ist. Bei CE geht alles per Hand. Zum Verständnis: im normalen VB kann man z.B. eine Textbox an eine bestimmte Datenbank andocken. Das ist einfach und bequem.

Für all diejenigen, die noch nie eine Datenbank programmiert haben, sollen hier ein paar Einstiegshilfen gegeben werden: Eine Datenbank ist eine strukturierte Datei, in der Daten eingelagert sind. Das Finden der Daten wird genauso strukturiert vorgenommen. Eine Datenbank hat Tabellen und diese wiederum hat ein oder mehrere Felder, in denen die Daten abgelagert werden.

Die Datenbank selbst hat natürlich einen Dateinamen. Bei CE ist die Endung .cdb. In unserem Falle wurde eine Datenbank namens sender.cdb angelegt. Diese Datenbank hat nur eine Tabelle namens ‚sender'. Die Tabelle wiederum hat 4 Felder: **nummer** (hier steht eine Zahl für den fortlaufenden Datensatz), das Feld **IDENT** enthält die CI-Daten der Antenne in hexadezimaler Form, das Feld **LA** (Location Aerea) enthält die entsprechenden LAI's in hexadezimaler Form, schließlich enthält das Feld **standort** die entsprechenden Beschreibungen des Antennenstandortes.

Beim Aufbau einer Datenbank ist es das A&O, dass man genau weiß, wie die einzelnen Informationen strukturiert gespeichert wurden.

Beim Sendersuchprogramm ist es wichtig, dass man ein Feld hat, in dem man per SQL die dazugehörigen Daten suchen kann. Jede Antenne hat eine eigene Identität, wie z.B. 03C7 im Hexformat. Dahinter versteckt sich der CellIDCode, wie oben beschrieben. Mit dem Wissen um den Namen der Zelle kann man jetzt in die Datenbank gehen. Welche weiteren Parameter gehören zu diesem Code? Man startet eine SQL-Anfrage mit der Identität, hier 03C7 z.B.. Wird der Datensatz gefunden, so sind alle anderen Parameter leicht zu ermitteln.

Das folgende Programm zeigt, wie man mit CE-Datenbanken umgehen muss. Eine ASCII-Datei wird hier in eine Datenbank umgewandelt, um später mit SQL-Befehlen relativ schnell suchen zu können. Wer noch nie mit Datenbanken zu tun hatte, der muss sich hier etwas einarbeiten. Hier wird zunächst gezeigt, wie man eine Datenbank mit Daten füllt.

Benutzt habe ich dazu eine ASCII-Datei namens cilalist.txt. Diese Liste fand ich bei www.duven.de , andere Listen für andere Bundesländer z.B. bei www.nobbi.com. Von hier aus findet man noch weitere Links zu anderen Sendersuchern.

Vielen Dank an die fleißigen Sendersucher, die dazu beigetragen haben, dass ich bisher noch in keiner unbekannten Zelle war.

Ca. 4400 Einträge findet man in der unteren Form vor, wovon ich hier nur zwei Einträge abgebildet habe. Die ersten 4 Bytes (000B) stehen für CI, die nächsten 4 Bytes stehen für LAI, die 26202 repräsentiert D2. Danach wird der Standort vermittelt. Der CI-Wert ist hier interessant. 000B ergibt dezimal 11. Es bedeutet, dass der Antennennamen daher DXB 1 und Segment 1 ist. Hier begann wohl alles mitten in Düsseldorf bei D2.

000B00D326202 Düsseldorf-Stadtmitte,Oststraße 86 / Friedrich-Ebert-Straße (Allianz-Gebäude)
000C00D326202 Düsseldorf-Stadtmitte,Oststraße 86 / Friedrich-Ebert-Straße (Allianz-Gebäude)

Die obige Datei habe ich ein wenig verändert, so dass der Standort besser dargestellt werden kann. Da zwischen den Angaben keine Leerzeichen sind, werden längere Einträge etwas komisch

dargestellt.

000B00D326202 Düsseldorf - Stadtmitte, Oststraße 86 / Friedrich-Ebert-Straße (Allianz-Gebäude)
000C00D326202 Düsseldorf - Stadtmitte, Oststraße 86 / Friedrich-Ebert-Straße (Allianz-Gebäude)

Das untenstehende Programm entält 5 Labels, einen Commandbutton. Kurz erklärt:
Rs wird als Recordset dimensioniert, conn als Connection.

1. Mit Set wird eine Referenz zu einem Object (Recordset.3.0 und Connection.3.0) hergestellt.
2. Rs wird geöffnet mit dem Befehl ‚CREATE DATABASE' und dem Pfadnamen + Name der Datenbank
3. Conn wird geöffnet mit dem Bezug zu der jetzt neu angelegten Datenbank
4. Rs wird geöffnet mit dem Befehl ‚CREATE TABLE'. Die Tabelle wird jetzt so eingerichtet, wie auch oben beschrieben.
5. Nummer wird als Integerzahl übergeben, IDENT und la sind immer 4 Bytes lang. Sie werden daher mit varchar (4) auf diese Länge festgelegt.
6. Da die Beschreibung in ihrer Länge variieren kann, wird sie als text festgelegt. Der Bezug zur Tabelle geht auf die Connection conn.
7. 1 und 3 sind Konstanten, die Eigenschaften der Datenbank beschreiben. Sie kann man in der Hilfe nachlesen.
8. Danach wird die Datenbank mit rs.open"sender",conn,1,3 geöffnet. Sie ist damit für Aktionen bereit.
9. Die Liste, in der die Senderdaten gespeichert sind wird geöffnet.
10. Danach wird über do while .... loop die gesamte Datei zeilenweise ausgelesen.
11.Ein Zähler läuft mit. Er liefert die Werte für das Feld ‚nummer'
12. Jetzt werden IDENT, LA und standort vom eingelesenen String abgeschnitten und in die entsprechenden Labels geschrieben.
13. Ist ein Datensatz komplett, so wird er abgespeichert.
14. Zunächst wird der Datenbankzeiger erhöht mit rs.addnew
15. Die Daten werden aus den Labels geholt und den Feldern zugewiesen.
16. Danach wird mit rs.update der Datensatz in der Datenbank aktualisiert. Er ist jetzt gespeichert.

Das ganze läuft in der Do While – Schleife und dauert ein paar Minuten.
Der gute Programmierer schließt sowohl die Datei, wie rs und conn.
Mit set=nothing wird der vorher aufgebaute Bezug zu dem Objekt wieder aufgehoben. (Hilft Speicher sparen)

```
Option Explicit

Dim rs As ADOCE.recordset
Dim conn As ADOCE.connection

Private Sub Command1_Click()
   Dim a, zaehler

   Set rs = CreateObject("ADOCE.Recordset.3.0")
   rs.Open "CREATE DATABASE '" & App.Path & "\sender.cdb'"

   Set conn = CreateObject("ADOCE.Connection.3.0")
   conn.Open App.Path + "\sender.cdb"

   Set rs = CreateObject("ADOCE.Recordset.3.0")
   rs.Open "CREATE TABLE Sender (nummer int,IDENT varchar(4),la varchar(4),standort text)", conn, 1, 3

   rs.Open "Sender", conn, 1, 3

 File1.Open App.Path + "\cilalist.txt", fsModeInput

Do While Not File1.EOF
   a = File1.LineInputString
   zaehler = zaehler + 1
         Label1 = zaehler
         Label2 = Mid(a, 1, 4)
         Label3 = Mid(a, 5, 4)
         Label4 = Mid(a, 9, 5)
```

```
        Label5 = Mid(a, 14)
    rs.AddNew
        rs.Fields("nummer") = CInt(Label1)
        rs.Fields("IDENT") = Label2
        rs.Fields("la") = Label3
        rs.Fields("standort") = Label5
    rs.Update

Loop

File1.Close
        rs.Close
        Set rs = Nothing
        conn.Close
        Set conn = Nothing
End Sub
```

Die notwendige Datenbank kann mit diesem Prograämmchen erstellt werden.

Jetzt folgt das Programm selbst, das sicherlich einige Programmiertricks aufweist, für die ich lange suchen musste. U.a. beschreibt es die Möglichkeit, über den Infrarotport seriell zu senden und auch zu empfangen. Normalerweise müsste man eigentlich nur eine Comm.ocx einsetzen, die den seriellen Datenverkehr regelt. Pustekuchen. Die kann zwar senden, aber ich - andere auch- habe noch nie etwas empfangen. Seriell über Kabel geht das wunderbar.

Die Lösung war nachher die API Creatfile zu benutzen und die gesendeten Bytes umzuwandeln: aus Unicode in Bytewerte. Es würde hier zu weit führen, dieses hier auszudiskutieren. Wer den Code sich ansieht, der wird sagen: was macht er denn jetzt?

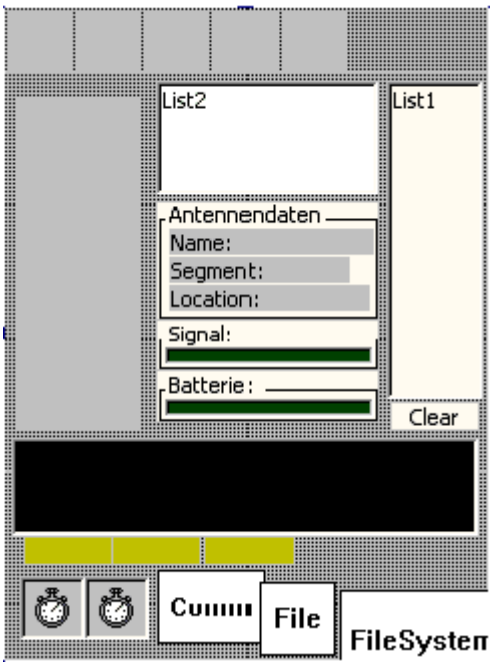
Wenn die Kommunikation über Kabel laufen soll, dann benutzt er ein Comm-Object. Richtig – mich selbst hat es auch verwundert. Trotz aller Versuche ist es mir nicht gelungen, die serielle Schnittstelle COM1 mit Createfile zu bedienen. Also diese Zwitterlösung. Arbeitet man mit Createfile, dann IR-Verbindung – ansonsten die COMM.OCX.

Da wir oben das Erstellen einer Datenbank beschrieben haben, müssen wir jetzt ja den umgekehrten Weeg gehen. Aus der Datenbank muss gelesen werden. Diese wenigen Programmzeilen sollen daher hier am Anfang diskutiert werden.

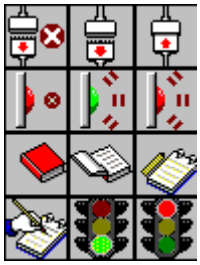
Die Prozedur mit conn und rs ist wieder die gleiche. Danach kommt ein sogenannter SQL-Befehl, der in der Datenbank nach einem speziellem Datensatz sucht.
Mit dem Antennenwert 0CD1 wie oben für die Variable **eintrag** wird die die Suche über „SELECT * FROM sender WHERE IDENT=“ & “0CD1“.conn,1,3 gestartet. Danach können die Feldeinträge abgefragt werden.

```
        Set conn = CreateObject("ADOCE.Connection.3.0")
        conn.Open App.Path + "\sender.cdb"

        Set rs = CreateObject("ADOCE.Recordset.3.0")
        rs.Open "Select * from Sender where IDENT=" & eintrag, conn, 1, 3

    Label1 = rs.fields(1).Value
    alttext = Label1
    Label2 = rs.fields(2).Value
    Label3 = rs.fields(0).Value
    Text2 = rs.fields(3).Value

    Label1.Visible = True
    Label2.Visible = True
    Label3.Visible = True

    rs.Close
    Set rs = Nothing
    conn.Close
    Set conn = Nothing
```

So sieht die Form zu dem Programm während der Entwicklung bei mir aus.



Es sind 8 Pictureboxen von pb1 bis pb8 zu laden., 2 Listboxen, ein Filesystem, eine Filecontrol, zwei Timer und einige Labels. Um nicht zu viele Files in einem Verzeichnis zu haben, habe ich die Bitmaps in einer Bitmap zusammengeführt (bis auf die Telefonbitmap SL45). Sie heißt sender.bmp. Wer es noch nicht so richtig ausprobiert hat: die einzelnen Bildelemente werden mit drawpicture herausgelesen.
Hier ein Beispiel: pb2.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 0, 32, 32
Es bedeutet, dass in der Picturebox namens pb2 ein Bild gezeichnet wird, das bei 0 horizontal, bei 0 vertikal beginnt und 32 Pixel horizontal breit ist und 32 Pixel vertikal hoch ist. Es wird ausgeschnitten aus dem Bild sender.bmp, das im Verzeichnis app.path liegt. Es ist der horizontale und vertikale Anfang (0,0) und (32,32) breit und hoch. Es handelt sich also um das Bild mit dem X am Kabel (links oben).



```
Option Explicit
Const OPEN_EXISTING = 3
Const CP_ACP = 0

Const EV_RXCHAR = &H1


Declare Sub Sleep Lib "Coredll" (ByVal dwMilliseconds As Long)

Declare Function CloseHandle Lib "Coredll" _
   (ByVal hObject As Long) As Long

Declare Function WaitCommEvent Lib "Coredll" _
(ByVal hFile As Long, lpEvtMask As Long, lpOverlapped As Long) As Long
```

```
Declare Function SetCommMask Lib "Coredll" _
   (ByVal hFile As Long, ByVal dwEvtMask As Long) As Long

Declare Function CreateFile Lib "Coredll" Alias "CreateFileW" _
   (ByVal lpFileName As String, _
   ByVal dwDesiredAccess As Long, _
   ByVal dwShareMode As Long, _
   lpSecurityAttributes As Long, _
   ByVal dwCreationDisposition As Long, _
   ByVal dwFlagsAndAttributes As Long, _
   ByVal hTemplateFile As Long) As Long

Declare Function WriteFile Lib "Coredll" _
   (ByVal hFile As Long, ByVal lpbuffer As String, _
   ByVal nNumberOfBytesToWrite As Long, _
   lpNumberOfBytesWritten As Long, _
   lpOverlapped As Long) As Long

Declare Function SetupComm Lib "Coredll" _
   (ByVal hFile As Long, ByVal dwInQueue As Long, ByVal dwOutQueue As Long) As Long

Declare Function WideCharToMultiByte Lib "Coredll" _
   (ByVal CodePage As Long, _
   ByVal dwFlags As Long, ByVal _
   lpWideCharStr As String, _
   ByVal cchWideChar As Long, _
   ByVal lpMultiByteStr As String, _
   ByVal cchMultiByte As Long, _
   ByVal lpDefaultChar As String, _
   ByVal lpUsedDefaultChar As Long) As Long

Declare Function SetCommTimeouts Lib "Coredll" _
   (ByVal hFile As Long, ByVal lpCommTimeouts As String) As Long

Declare Function ReadFile Lib "Coredll" _
   (ByVal hFile As Long, _
   lpbuffer As Byte, _
   ByVal nNumberOfBytesToRead As Long, _
   lpNumberOfBytesRead As Long, _
   lpOverlapped As Long) As Long

   Dim buff As String
   Dim irda, speichern, altbatt, altdbm, alttext
   Dim conn As ADOCE.Connection
   Dim rs As ADOCE.Recordset
   Dim ampel, buch, buchs, versuch
Private Sub senden(schreibew)
Dim r
Dim geschrieben
Dim schreibeA As String

   r = WideCharToMultiByte(CP_ACP, 0, schreibew, -1, schreibeA, 255, 0, 0)
   r = WriteFile(irda, schreibeA, r, geschrieben, 0)
End Sub
Private Sub empfangen()
Dim r, rr, mask, ret
buff = ""
Dim bu As Byte
   Do
      rr = WaitCommEvent(irda, mask, 0)
      versuch = versuch + 1
      Caption = versuch
```

```
      If versuch > 100 Then

         versuch = 0
         Timer1.Enabled = False
         If irda Then ret = CloseHandle(irda)
         pb3.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 32, 32, 32

      Exit Sub
      End If
   Loop While Not ((mask And EV_RXCHAR) = EV_RXCHAR)

   versuch = 0

   Do
      rr = ReadFile(irda, bu, 1, r, 0)
      buff = buff + Chr(bu)
   Loop While (r = 1)
End Sub


Private Sub Comm1_OnComm()
buff = buff + Comm1.Input
End Sub




Private Sub Form_Load()
pb1.Picture = App.Path & "\sl45.bmp"
pb2.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 0, 32, 32
pb3.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 32, 32, 32
pb4.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 64, 64, 32, 32
pb6.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 64, 32, 32
pb8.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 64, 96, 32, 32
ampel = False
alttext = "Test"

Label1.Left = pb1.Left + 9
Label1.Top = pb1.Top + 54
Label1.ZOrder 0
Label1.FontBold = True

Label2.Left = Label1.Left
Label2.Top = Label1.Top + Label1.Height
Label2.ZOrder 0
Label2.FontBold = True

Label3.Left = Label1.Left
Label3.Top = Label2.Top + Label2.Height
Label3.ZOrder 0
Label3.FontBold = True
End Sub

Private Sub Form_OKClick()
   App.End
End Sub

Private Sub Label5_Click()
List1.Clear
End Sub

Private Sub List1_Click()
Call Datenbank_durchsuchen(List1.List(List1.ListIndex))
```

```
End Sub

Private Sub List2_Click()
File1.Open App.Path + "\" + List2.List(List2.ListIndex), fsModeInput
List1.Clear
   Do While Not File1.EOF
      List1.AddItem File1.LineInputString
   Loop

File1.Close
Label5 = List1.ListCount
List1.Visible = True
List2.Visible = False
Text2.Visible = True
pb6.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 64, 32, 32
End Sub

Private Sub pb2_Click()
   Timer1.Enabled = False
      pb2.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 0, 32, 32
      pb3.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 32, 32, 32
   If Comm1.PortOpen = False Then Comm1.PortOpen = True
      Timer2.Enabled = Not Timer2.Enabled
      If Timer2.Enabled = True Then
         pb8.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 96, 32, 32
      Else
         pb8.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 64, 96, 32, 32
      End If
End Sub

Private Sub pb3_Click()
Dim ret
   Timer2.Enabled = False

   If irda Then ret = CloseHandle(irda)

   irda = CreateFile("COM3:", 0, 0, 0, OPEN_EXISTING, 0, 0)  'COM Öffnen.
   ret = SetCommMask(irda, EV_RXCHAR)

   buff = ""
   Timer1.Enabled = Not Timer1.Enabled
      If Timer1.Enabled = True Then
      pb3.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 32, 32, 32
      pb8.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 96, 32, 32
   Else
      pb8.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 96, 96, 32, 32
      pb3.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 32, 32, 32
   End If
End Sub

Private Sub pb4_Click()
If Not buchs Then
pb4.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 64, 64, 32, 32
speichern = True
buchs = True
Else
pb4.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 96, 32, 32
speichern = False
buchs = False
End If
End Sub

Private Sub pb6_Click()
```

```
Dim exist

If Not buch Then
   List2.Clear
   pb6.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 64, 32, 32
   exist = FileSystem1.Dir(App.Path + "\*.txt")
  Do While Len(exist)
   List2.AddItem exist
   exist = FileSystem1.Dir
  Loop
 List2.Visible = True
   buch = True
Else
   pb6.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 0, 64, 32, 32
   List2.Visible = False
   buch = False
End If
End Sub
Private Sub Form_OKClick()
   App.End
End Sub


Private Sub PictureBox1_Click()
If ampel Then
   pb7.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 96, 32, 32
   Else
   pb7.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 64, 96, 32, 32
End If
End Sub

Private Sub pb7_Click()

End Sub

Private Sub pb8_Click()
If ampel Then
   pb8.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 96, 32, 32
   ampel = False
Else
   pb8.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 64, 96, 32, 32
   ampel = True
End If
End Sub

Private Sub Timer1_Timer()
If ampel Then Exit Sub

   If pb3.BackColor = vbGreen Then pb3.BackColor = vbRed Else pb3.BackColor = vbGreen

   If pb3.BackColor = vbGreen Then
      pb3.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 32, 32, 32
   Else
      pb3.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 64, 32, 32, 32
   End If
   If irda <= 0 Then Exit Sub

   Call senden("at+creg?" + vbCr)
      Sleep 200
      Call empfangen
      Call Eintrag_checken(buff)
      Call senden("at+csq" + vbCr)
      Sleep 200
```

```
      Call empfangen
   If InStr(buff, "+CSQ:") Then Call Sender_staerke
      Call senden("at+cbc" + vbCr)
      Sleep 200
      Call empfangen
      If InStr(buff, "+CBC:") Then Call Batterie_checken
End Sub

Private Sub Timer2_Timer()
If ampel Then Exit Sub
If pb2.BackColor = vbRed Then pb2.BackColor = vbGreen Else pb2.BackColor = vbRed
   If pb2.BackColor = vbRed Then
      pb2.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 32, 0, 32, 32
   Else
      pb2.DrawPicture App.Path + "\sender.bmp", 0, 0, 32, 32, 64, 0, 32, 32
   End If
   Comm1.Output = "at+creg?" + vbCr
      Sleep 200
   Comm1.Output = "at+csq" + vbCr
      Sleep 200
   Comm1.Output = "at+cbc" + vbCr
      Sleep 200
   If InStr(buff, "+CSQ:") Then Call Sender_staerke
   If InStr(buff, "+CBC:") Then Call Batterie_checken

Call Eintrag_checken(buff)

End Sub
Private Sub Batterie_checken()
Dim i, batt

   batt = Mid(buff, InStr(buff, "+CBC:") + 8)
   batt = Mid(batt, 1, InStr(batt, Chr(13)) - 1)
   If altbatt = batt Then Exit Sub
      altbatt = batt
   Frame2.Caption = "Batterie: " & batt & "%"

   pb7.DrawLine 0, 0, pb7.Width / 100 * (batt / 100 * 100), pb5.Height, vbGreen, True, True
      For i = pb7.Width / 10 To pb7.Width Step pb7.Width / 10
         pb7.DrawLine i, 0, i, pb7.Height, vbYellow
      Next i
   Frame2.Visible = True
End Sub
Private Sub Sender_staerke()
Dim teil1, signal, i
   teil1 = Mid(buff, InStr(buff, "+CSQ:") + 6)
   signal = CInt(Mid(teil1, 1, InStr(teil1, ",") - 1))

   If altdbm = signal Then Exit Sub
      altdbm = signal
      Frame1.Visible = True
      Frame1.Caption = "Signal: " & -(113 - signal * 2) & " dBm"
      pb5.Cls
      pb5.DrawLine 0, 0, pb5.Width / 100 * (signal / 31 * 100), pb5.Height, vbGreen, True, True

   For i = pb5.Width / 12 To pb5.Width Step pb5.Width / 12
      pb5.DrawLine i, 0, i, pb5.Height, vbYellow
   Next i
End Sub

Private Sub Eintrag_checken(eintrag)
On Error Resume Next
Dim i, vorhanden
```

```
eintrag = Mid(eintrag, InStr(eintrag, "+") + 1)
eintrag = Mid(eintrag, InStr(eintrag, Chr(34) + "," + Chr(34)) + 3)
Label1 = Mid(eintrag, 1, InStr(eintrag, Chr(34)) - 1)

buff = ""
   For i = 0 To List1.ListCount - 1
      If List1.List(i) = Label1 Then vorhanden = True: Exit For
   Next i
   If vorhanden = False Then
      If Label1 <> "" Then
         List1.AddItem Label1.Caption
         Label5 = List1.ListCount
         If speichern Then
            File1.Open App.Path & "\S" & Date & ".txt", fsModeAppend
            File1.LinePrint Label1
            File1.Close
         End If
      List1.Visible = True
      Text2.Visible = True
      End If

   End If

If Label1 <> alttext Then Call Datenbank_durchsuchen(Label1.Caption)
End Sub
Private Sub Datenbank_durchsuchen(eintrag)
Dim antenne, ausgabe

Set conn = CreateObject("ADOCE.Connection.3.0")
   conn.Open App.Path + "\sender.cdb"
Set rs = CreateObject("ADOCE.Recordset.3.0")
   rs.Open "Select * from Sender where IDENT=" & eintrag, conn, 1, 3

   Label1 = rs.fields(1).Value
   alttext = Label1
   Label2 = rs.fields(2).Value
   Label3 = rs.fields(0).Value
   Text2 = rs.fields(3).Value

   Label1.Visible = True
   Label2.Visible = True
   Label3.Visible = True

   rs.Close
   Set rs = Nothing
   conn.Close
   Set conn = Nothing


   antenne = CStr(CInt("&H" & eintrag))

   Label4 = "Name: DXB " + Mid(antenne, 1, Len(antenne) - 1)
   Label6 = "Segment: " + Mid(antenne, Len(antenne))
   Label7 = "Location: " & CInt("&H" & Label2)
   Frame3.Visible = True

End Sub
```

Das Programm ist bei weitem noch nicht optimiert. Da es wahrscheinlich auch keine Massenverbreitung aufweisen wird, kann der eine oder andere noch darin optimieren. Mir hat es jedenfalls Spaß gemacht, etwas über die Technik von GSM zu erfahren. Mittlerweile kenne ich alle Sendemasten, die ich morgens und abends auf dem Weg zum WDR und zurück durchfahre. Es ist

nicht immer die gleiche Zahl. Es schwankt zwischen 20 und 24 – je nachdem, wie ausgelastet gerade eine Antenne ist.

Zu den abgebildeten Icons sollte ich noch ein paar Worte verlieren, was ich mir dabei gedacht habe.

Das 1. Icon von links stellt dar:
Wird es angeklickt, so wünscht man eine Kabelverbindung über COM1 zu dem Handy. Das Wechseln des Icons zeigt an, dass diese Funktion aktiviert wurde.
Das zweite Icon von links stellt dar:
Wird es angeklickt, so wünscht man eine Infrarotverbindung über COM3 zu dem Handy. Das Wechseln des Icons zeigt an, dass diese Funktion aktiviert wurde. Es kann durchaus problematisch sein, die Verbindung zu installieren. Manchmal musste ich es 5 mal probieren, manchmal klappte es sofort. Manchmal sogar musste ich das Handy ausschalten und neu booten.
Das dritte Icon von links stellt dar:
Die empfangenen Werte werden zwar angezeigt – sie werden aber nicht gespeichert. Wird dieses Icon aktiviert, so wird eine Datei geöffnet, die den Namen S+datum erhält, z.B. S12.11.01.txt. Es handelt sich dann um einen Mitschnitt vom 12. November 2001. Zuhause am PC kann man dann die Datei entsprechend umbenennen.
Das vierte Icon von links stellt dar:

Hier kann man die abgespeicherten Daten einlesen. Ist aktiver Empfang von Zellinformationen jedoch eingeschaltet, so werden die Standortinformationen überschrieben von den aktuell empfangenen Daten.
Um dies zu verhindern kann man das Ampelicon anklicken. Zeigt das Symbol rot, so ist die Aktualisierung außer Kraft gesetzt, man kann jetzt abgespeicherte Routen anzeigen.

Wir sind hiermit eigentlich noch lange nicht am Ende des Sendersuch- Hobbies. Es gehört dazu, dass man neue oder unbekannte Antennen aus seiner Umgebung den Sendersuchern meldet, die die Information dann wieder im Netz zur Verfügung stellen. Wer aber eine Antenne melden will, der muss genau wissen, um welchen Standort es sich handelt. Hat man eine CI-Information, die noch nicht in der Datenbank gespeichert wurde, so muss man noch weiter ermitteln: „Wo ist die Antenne?“, „kann man sie sehen und fotografieren?“. Es gibt weitere Informationen, die hier nicht umgesetzt wurden. Wird z.B. ei Gespräch geführt, so wird das Handy angewiesen, bestimmte Zeitslots einzuhalten. Diese Zeitslots werden entfernungsabhängig vergeben. Wenn man die eingestellten Zeitslots kennt, so kann man rückschließen, wie weit die Antenne vom Handy entfernt ist. Empfängt man den Wert 5 für TA, so bedeutet dies: die Antenne ist ca. 5 * 550 bis 6 *550 Meter weit weg. Jetzt kann man suchen und eindeutig bestimmen.