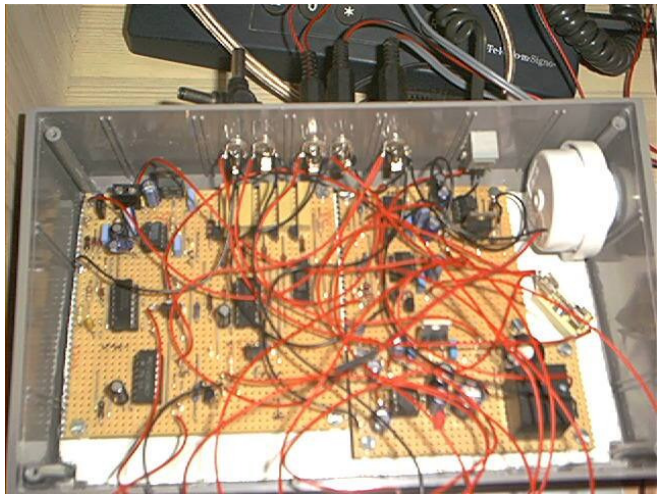
5-Tonfolgen Dekodierung mittels AVR

2. Fassung

In letzter Zeit bekam ich viele Anfragen wie denn die Dekodierung der Tonfolgen über einen µC genau funktioniert, wie die Hardware und Software dazu aussieht und was das ganze kostet. Ich möchte nun an dieser Stelle meine Versuche zusammenzufassen und hoffe dass auch einige einen Nachbau wagen und mir über die Ergebnisse berichten.

Vorgeschichte:

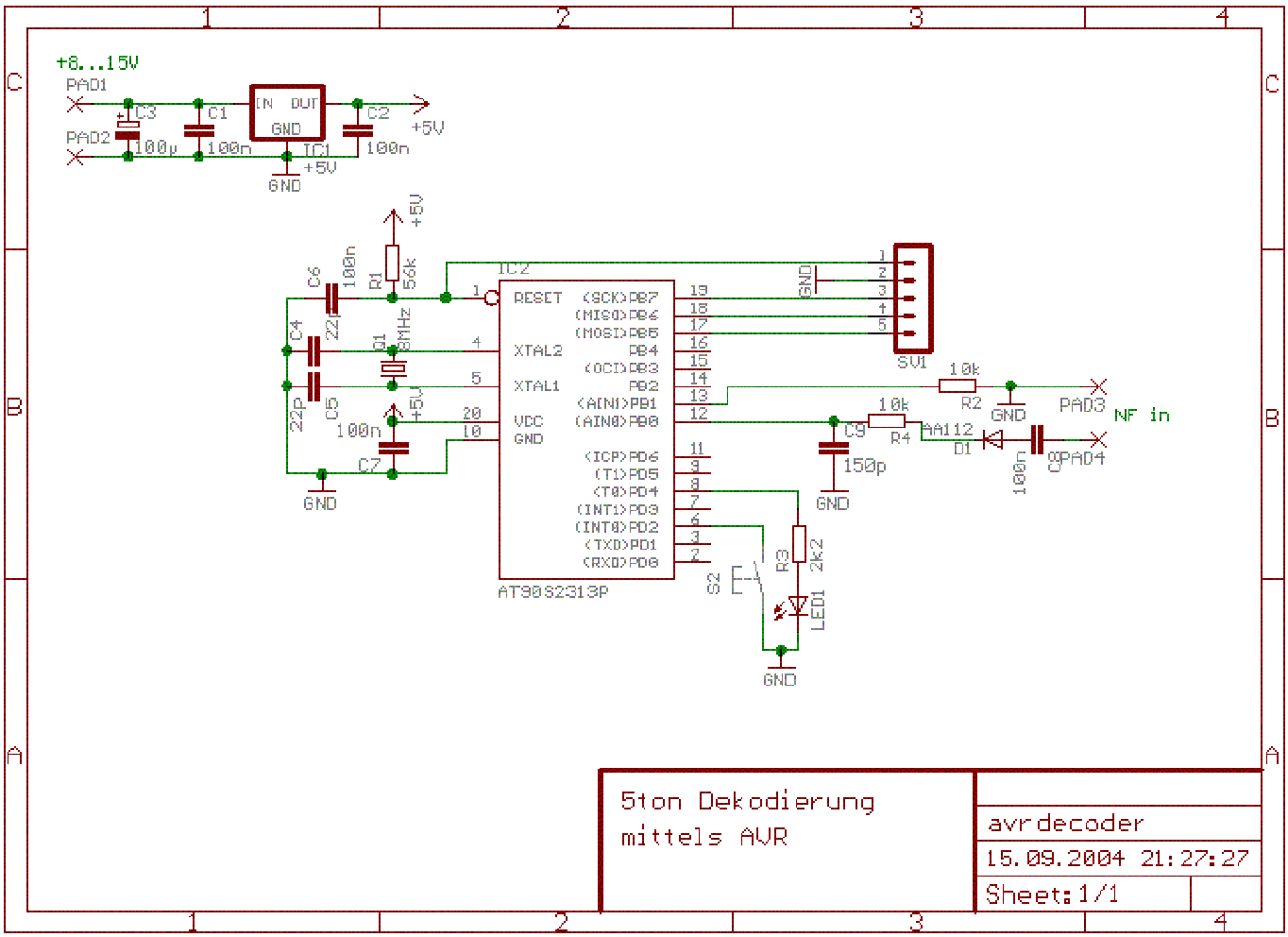
Ich beschäftige mich mit der Tonfolgen Dekodierung schon seit 2 Jahren. Als Mitglied einer kleinen freiwilligen Feuerwehr habe ich das Problem die Sirene kaum zu hören. Da ich ein bisschen Elektronik begeistert bin kam ich schnell auf die Idee mir eine eigene Alarmierung zu bauen. Lösungen über den PC wollte ich von Anfang an ausschließen. Ich suchte also nach Schaltplänen im Web und stieß auf einen recht Interessanten Link:
http://www.ginko.de/user/flasche/familie/5tondec.html
Dies ist ein relativ aufwändiger Hardwaredekoder den ich allerdings nicht so aufbaute, weil mich schon einmal die doppelseitige Platine abschreckte. Ich kaufte mir also den NE567 und begann auf dem Steckbrett einen eigenen Entwurf, der vom Prinzip her genauso arbeitet wie der Dekoder im oben genannten Link. Die Schaltung funktionierte dann auch wunderbar und versieht auch heute noch ihren Dienst. Sie hat noch nicht einmal versagt! Leider reicht ja ein reiner Dekoder nicht aus um eine Alarmierung zu bewerkstelligen, es kamen also noch ein paar Monoflops und Flipflops dazu um einen reellen Betrieb zu ermöglichen. Mittlerweile war der Schaltungsaufwand schon relativ groß und schwer zu reproduzieren. Als nächstes kam mir die Idee in Falle eines Alarmes mein Handy anrufen zu lassen. Dies geschieht über ein umgebautes Telefon, dass über den Babyruf einer Telefonanlage anruft und die NF des Scanners überträgt. So kann ich die Alarmmeldung am Handy mithören. Das funktioniert auch alles wunderbar, doch mittlerweile ist die Schaltung dermaßen unübersichtlich und aufwändig, dass ich über eine Vereinfachung nachdachte. So sieht mein Aufbau aktuell aus, der AVR ist noch nicht im Einsatz:



Wie gesagt, so funktioniert alles einwandfrei, aber viel zu kompliziert für einen Nachbau.
Ich fing dann schließlich an mich mit Mikroprozessoren zu beschäftigen speziell mit den AVRs von Atmel (www.atmel.com). Bald kam mir die Idee die Dekodierung über einen AVR zu lösen. Also schnell mal informiert wie die Möglichkeiten so aussehen. Es gibt fertige ICs, die

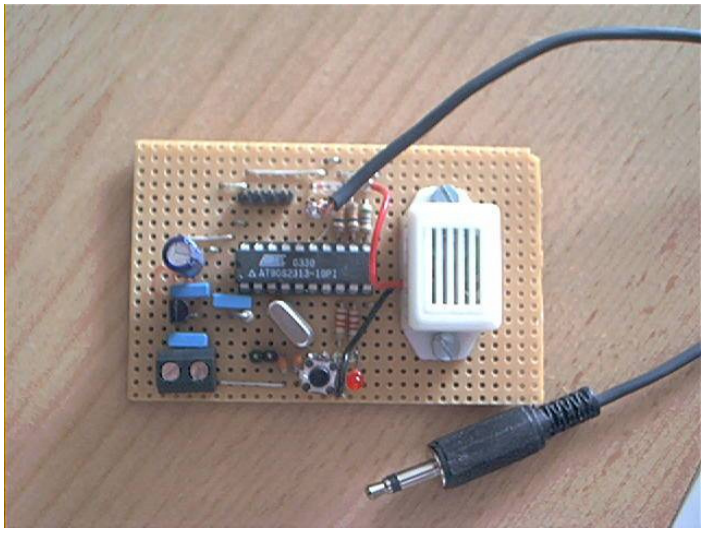
5Tonfolgen Dekodieren können, z.B. den CMX813 von CML (http://www.alldatasheet.com/datasheet-pdf/pdf/CMLMICRO/CMX813/datasheet.pdf)
Die Dinger sind eigentlich optimal, man müsste sich um nicht kümmern und hätte den µC für andere Sachen frei. Das Problem hierbei ist, dass diese ICs sehr teuer und schwer zu kriegen sind. Also auch nichts ☹. Ich überlegte, ob es nicht ein AVR auch direkt schaffen kann die Tonfolgen zu dekodieren. Eine Frequenz genau genug einzulesen ist kein Problem über den in vielen AVRs vorhanden 16Bit breiten Zähler. Nur ob das auch schnell genug möglich ist??
Nach einer Nachfrage in einem Elektronik Forum (unter www.spotlight.de) bestätigte man mir, dass es gehen müsste und gab mir auch einen sehr hilfreichen Tipp, wie man softwaremäßig am besten die Frequenz auswertet. Den Rest der Geschichte findet ihr in der Schaltungsbeschreibung.

Minimaler Hardwareaufbau:



Dies ist die einfachste und günstigste Art eines Dekoders. So wie er im Schaltplan dargestellt ist kann er jede beliebige 5-Tonfolge dekodieren und per UART (serielle Schnittstelle) ausgeben. Zusätzlich kann bei einer bestimmten Folge z.B eine LED oder ein Summer aktiviert werden. Der Taster dient zum Reset des Alarmes. Viel mehr Spielraum besteht aufgrund des beschränkten Programmspeichers leider nicht mehr. Alle Teile die zum Aufbau sind z.B bei Reichelt erhältlich. Preislich dürfte die reine Schaltung ungefähr bei 5 Euro liegen. Die Schaltung funktioniert wirklich sehr zuverlässig und kann also einen fertigen Dekoder IC vollständig ersetzen.

Hier ein Bild des Aufbaus auf Streifenraster:



Grobe Beschreibung der Schaltung:

Das NF-Signal vom Funkempfänger wird an Pad4 eingespeist. Die Germaniumdiode (wegen geringem Spannungsabfall) richtet das Signal gleich. Der 16bit Counter des AT90S2313 wird im caputure mode betrieben, dass bedeutet, dass er bei jedem Quarztakt um 1 hochzählt. Wenn der dem Counter vorgeschaltete analog Komparator bei einer neuen Periode des NF-Signals das Capture Ereignis auslöst wird ein Interrupt ausgelöst und der Zählerstand gesichert. Beim zweiten mal wird der alte Zählerstand mit dem neuen verglichen und aus $\frac{Zähler_{neu} - Zähler_{alt}}{Quarzfrequenz}$ lässt sich die aktuelle Frequenz der NF-Signals berechnen. Diese wird mit jeder Periode des NF-Signals neu bestimmt und dient als Grundlage für die Dekodierung eines Tones bzw. der gesamten Folge.
Der Zähler wird mit der Quarzfrequenz von 8MHz getaktet. Wenn man ein bisschen rechnet kann man also ohne Überlauf bis ca. 130 Hz runterzählen und das fast aufs Hz genau.

Die Software zur oben gezeigten Schaltung:

Ich habe den AVR in Bascom programmiert, das ist eine einfache sehr stark an Basic angelehnte Programmiersprache. Auf www.mcselec.com gibt es eine voll funktionsfähige Demoversion des Compilers, mit der auch der µC gebrannt werden kann. Wer sich einmal ein bisschen über AVRs mit Bascom informieren will schaut am besten mal bei www.rowalt.de vorbei. Dort gibt's ein kleines Tutorial über Bascom in dem auch erklärt wird wie man den µC programmiert. Hier ist aber erst einmal der Quellcode der Software:

```
$regfile = "2313def.dat"
$crystal = 8000000


Ddrd = &B00010000
Portd.2 = 1


Config Timer1 = Timer , Prescale = 1 , Capture Edge = Rising , Noise Cancel = 1
Config Aci = On , Compare = On , Trigger = Rising

On Icp1 Oncapture
Enable Icp1

On Int0 Ontaster
Config Int0 = Falling
Enable Int0

Enable Interrupts

Dim Zaehler As Word
Dim Icr_neu As Word
Dim Icr_alt As Word
Dim Frequlong As Long
Dim Frequenz As Word
Dim A(5) As Word
```

```
Dim Frequsumme As Word
Dim Adizaehler As Integer
Dim Durchschnitt As Word
Dim Grenzeo As Word
Dim Grenzeu As Word
Dim Ton As String * 1
Dim Tonalt As String * 1
Dim Folge As String * 5
Dim Timeout As Word


Main:


Waitms 2
Frequlong = 8000000 / Icr_neu
Icr_neu = 1
Frequenz = Frequlong

Incr Timeout
If Timeout > 40 Then
  Folge = ""
  Tonalt = ""
  Timeout = 0
End If


A(5) = A(4)
A(4) = A(3)
A(3) = A(2)
A(2) = A(1)
A(1) = Frequenz

Frequsumme = 0
For Adizaehler = 1 To 5
  Frequsumme = Frequsumme + A(adizaehler)
Next

Durchschnitt = Frequsumme / 5
Grenzeo = A(1) + 10
Grenzeu = A(1) - 10

If Durchschnitt > Grenzeu And Durchschnitt < Grenzeo Then

  Select Case Durchschnitt
    Case 1050 To 1070 : Ton = "1"
    Case 1150 To 1170 : Ton = "2"
    Case 1260 To 1280 : Ton = "3"
    Case 1390 To 1410 : Ton = "4"
    Case 1520 To 1540 : Ton = "5"
    Case 1660 To 1680 : Ton = "6"
    Case 1820 To 1840 : Ton = "7"
    Case 1990 To 2010 : Ton = "8"
    Case 2190 To 2210 : Ton = "9"
    Case 2390 To 2410 : Ton = "0"
    Case 2590 To 2610 : Ton = "w"
    Case Else : Goto Main
  End Select

Else : Goto Main
End If

If Ton = Tonalt Then Goto Main
Tonalt = Ton

Folge = Folge + Ton
Timeout = 0
If Len(folge) = 5 Then
  Print Folge
  If Folge = "2910w" Then
    Portd.4 = 1
  End If
  Folge = ""
End If

Goto Main
```

```
Ontaster:
Portd.4 = 0
Return


Oncapture:
  Zaehler = Timer1
  Icr_neu = Zaehler - Icr_alt
  Icr_alt = Zaehler
Return
```

OK, das wäre nun die Software, sieht doch ganz einfach aus, oder ☺?
Ich habe sie leider nicht kommentiert, deswegen ein paar Erläuterungen dazu:

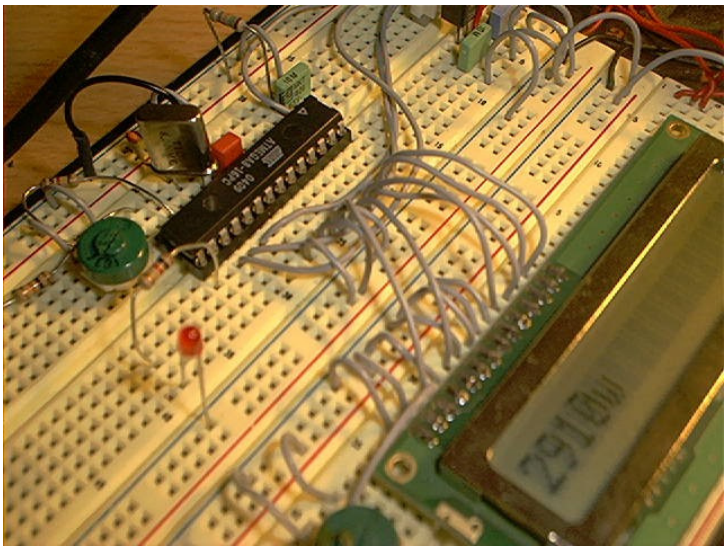
Fangen wir ganz unten mit der Interruptroutine „Oncapture“ an: Hier wird immer die aktuelle Zählerdifferenz ermittelt, sie steht immer dem Programm in der Variablen „Icr_neu“ (Icr für Input Capture Register) zur Verfügung. Am Anfang des Hauptprogramms (nach Main) wird erst einmal aus dem Zählerstand die aktuelle Frequenz berechnet, mit der weiter gearbeitet wird. Die Frequenz wird dann später duch ein Array „durchgeschoben“, so wird ermittelt, ob sie über einen längeren Zeitraum konstant ist, also ob ein Ton anliegt. Es wird der Durchschnitt des Arrays berechnet und wenn dieser Durchschnitt ungefähr mit der aktuellen Frequenz übereinstimmt weiß das Prog, dass ein Ton vorhanden ist. Wenn nicht springt es zurück an den Anfang (Main). Bei einem erkannten Ton wird in der Select Case Anweißung geschaut ob der Ton in die ZVEI Reihe passt. Wenn nicht der selbe Ton vorher schon einmal kam wird er der String Variablen „Folge“ hinzugefügt. Wenn die Folge 5 Zeichen lang ist wird sie ausgegeben und mit der „Alarmfolge“ verglichen. Wenn sie passt werden die Ports angeschaltet, an denen z.B ein Summer hängt. In der Interruptroutine „Ontaster“ werden diese Ports bei Tastendruck wieder ausgeschaltet. Die Variable „Timeout“ dient dazu um die Variable „Folge“ bei unvollständigen Folgen oder falsch erkanten Tönen zurückzusetzen.
So, das wars eigentlich schon. Wer mehr wissen will schaut am besten in das oben genannte Tutorial. Damit lassen sich fast alle Fragen zum Quelltext klären.

Ich will diese Schaltung nachbauen, aber wie??

Die Software passt zur oben gezeigten Schaltung. Wenn ihr euch Bascom besorgt könnt ihr den Quelltext dort einfügen ihn compillieren und über SV1 den µC ISP mithilfe der parallelen Schnitstelle brennen. Wie das genau geht erfahrt ihr im oben genannten Tutorial. Anpassungen an eigene Vorstellungen sind dann auch ganz einfach möglich. Wie schon gesagt ist der Programmspeicher des 2313 fast voll belegt mit diesem Programm, man kann aber z.B den Print Befehl rausschmeißen, dann gibt’s wieder ein bisschen Platz. Eine andere Möglichkeit ist natürlich auf einen anderen Prozessor auszuweichen, z.B. den Atmega8, wie ich es zur Zeit für meine Versuche tue.

Noch etwas Grundlegendes:
Wer noch nie vorher mit µCs gearbeitet hat wird sich schwer tun dieses Projekt zu realisieren. Auch wenn es nicht besonders anspruchsvoll ist gibt es als Anfänger einfach zu viele Stolpersteine.
Ich kann hier nur wieder auf www.rowalt.de verweißen. Mit diesem Tutorial habe ich den Einstieg in die Spannende Welt der AVR geschafft.

Wie geht es weiter?



So sieht mein aktueller Aufbau auf dem Steckbrett mit dem Atmega8 aus. Mittlerweile ist auch die Telefonwahl über den Befehl dtmfout in Bascom realisiert. Sie muss aber noch ein bisschen verbessert werden.

Also, wer irgendwelche Versuche unternimmt das nachzubauen kann sich ja mal bei mir melden, ich würde mich freuen. Auch für sonstige Anregungen bin ich immer offen:

Mail: martinhaun@freenet.de
ICQ: 117-553-521