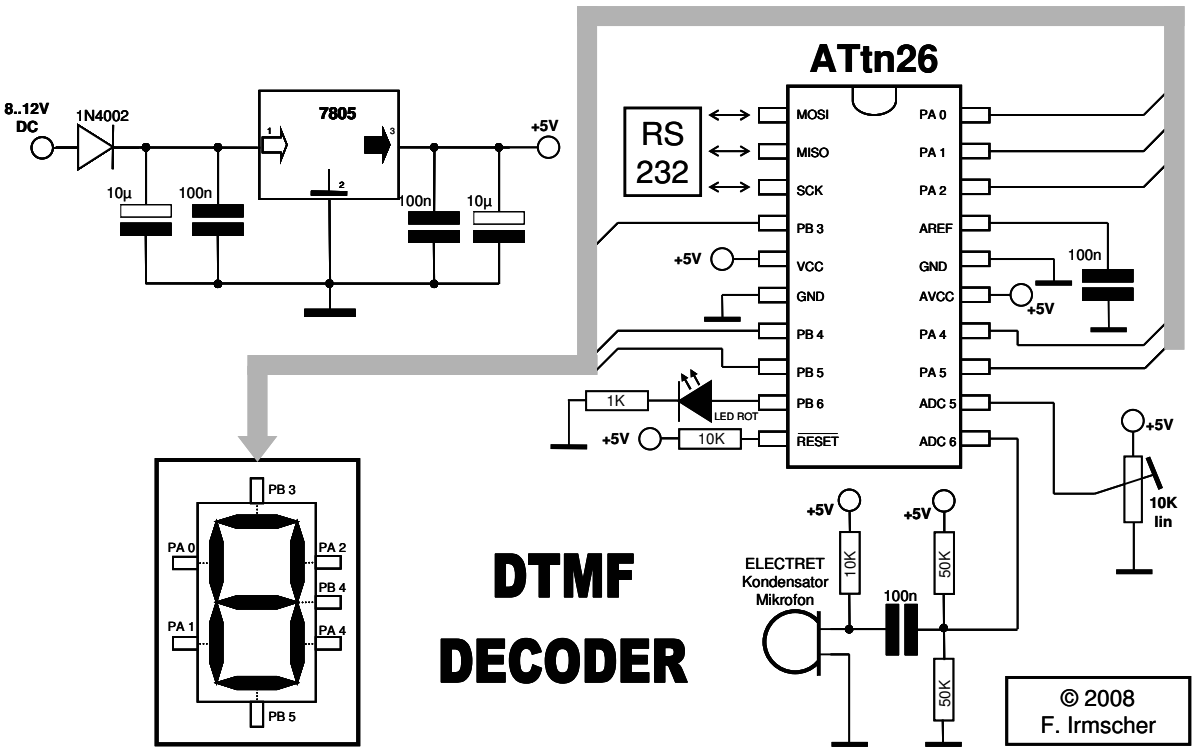
DTMF - DECODER

- für ATtiny 26
- ohne FFT und Goertzel – Algorithmus!
- Analyse der Interferenzmuster
- 16-bit Frequenzzähler



Einleitung

Das sog. ***MVF (Mehrfrequenzverfahren)*** oder auch ***DTMF (Dual Tone Multiple Frequency)*** wird bei der Telefonvermittlungstechnik genutzt. Es wird gebildet durch eine Überlagerung von zwei sinusförmigen Tonsignalen. Bei der Dekodierung gilt es, die beiden Frequenzbestandteile zu erkennen, um daraus die zugehörigen Tastencodes zu identifizieren:

	1209 Hz	**1336 Hz**	**1477 Hz**
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

Weitere Informationen sind zum Beispiel bei Wikipedia zu finden.

Zur Dekodierung werden meist spezielle Rechenoperationen verwendet: Die sog. Fast-Fourier-Transformation (FFT) oder der Goertzel-Algorithmus. Hiermit kann präzise Frequenzanalyse betrieben werden. Sie erfordern jedoch sehr viel Speicherplatz. Außerdem werden Kenntnisse in höherer Mathematik benötigt.

Mein Interesse bestand darin, für einen Mikrocontroller der Tiny-Klasse, hier den **ATtiny26,** einen alternativen Weg zu finden, um DTMF-Codes zu entschlüsseln.
Es werden dabei Ausschnitte der **Interferenzmuster** der beiden sich überlappenden Frequenzen analysiert. Ausgabe erfolgt mit einer Sieben-Segment-Anzeige. Durch das spezielle Vorgehen kann auf ein Quarz verzichtet werden. Es reicht die voreingestellte Taktfrequenz von nur 1MHz aus. Somit kann auch einfach mit dem ***Lernpaket Mikrocontroller*** programmiert werden!
Für das Electret-Kondensator-Mikrofon wird kein zusätzlicher Operationsverstärker benötigt, die Verstärkung erfolgt über ein sog. *Differential ADC Channel Pair with gain* (x20) des ATtiny26.
Fazit:
Ein DTMF-Decoder wird auf diese Weise realisierbar. Die Schaltung benötigt jedoch ein längeres Signal zur Analyse (ca. 500ms).
Vielleicht interessiert Sie aber auch einfach nur der ungewöhnliche Lösungsweg?
Auf die Funktionsfähigkeit eines Nachbaues wird keine Gewährleistung gegeben!

Anleitung

Programmierung

Der ATtiny26 kann z.B. mit dem Programmiermodul des ***Lernpaket Mikrocontroller*** von Franzis programmiert werden. Das Vorgehen ist an anderer Stelle beschrieben. Die Fuses müssen dabei nicht verändert werden.

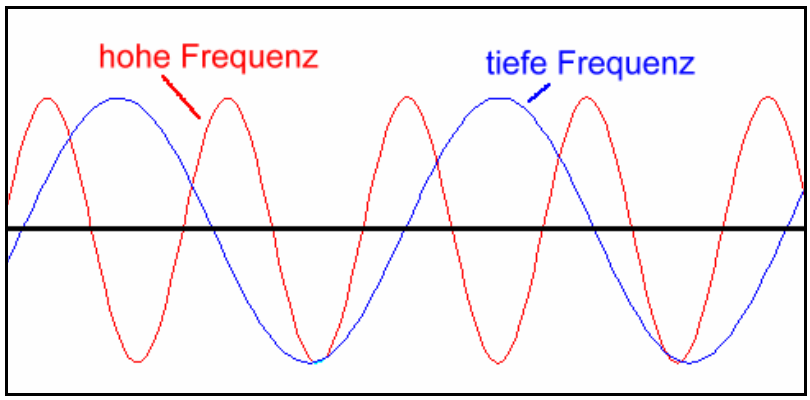
Elektronik

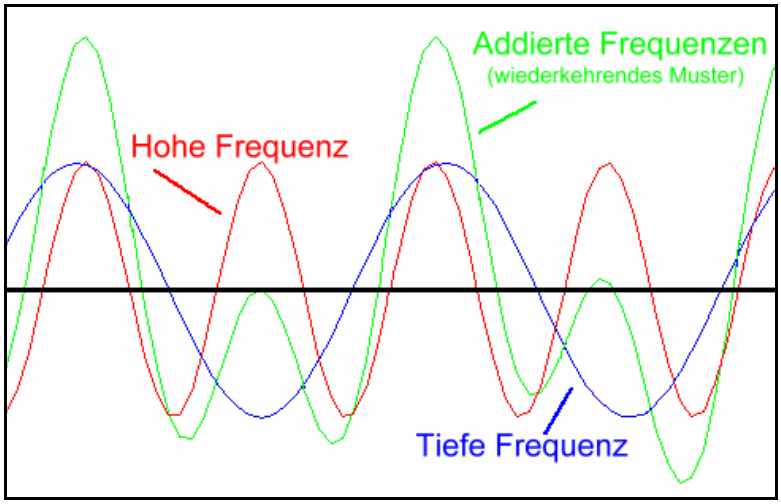
Es werden nur sehr wenig externe Bauelemente benötigt. Electret-Kondensator-Mikrofone enthalten bereits einen schwachen, für diese Zwecke ausreichenden Vorverstärker.
Als Betriebsspannung sollen ca. 9V angelegt werden. Diese Spannung wird durch den Spannungsregler auf stabilisierte 5V heruntergeregelt.
Die LED dient zur optimalen Arbeitspunkteinstellung. Zunächst wird ein lautes akustisches Signal auf das Mikrofon gerichtet. Wird der Trimmer weit nach rechts oder links gedreht, leuchtet die LED auf. Der Trimmer ist so zu justieren, dass die LED nicht leuchtet und vom rechten und linken Leuchtbeginn gleich weit entfernt ist. Leuchtet die LED auch in Mittelstellung, so ist das Signal zu stark!

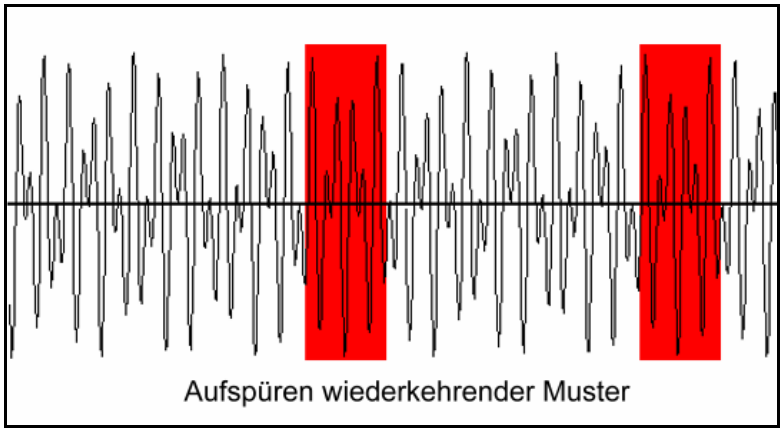
Programmbeschreibung

Die Idee

Das DTMF-Signal besteht aus zwei sich überlappenden Sinus-Wellen bekannter Frequenzen.
Erzeugt man diese Signale und betrachtet sie mit dem Oszilloskop, so kann man erkennen, dass durch Addition der beiden Signale ein für jeden Code typisches Frequenzmuster entsteht.







Dabei ist zu erkennen, dass die höhere der beiden Frequenzen als Welligkeit im Signal stets erhalten bleibt, wenn auch mit zum Teil stark schwankender Amplitude. Die sichtbare Wellenlänge variiert durch die Interferenz geringfügig. Im Mittel über mehrere Perioden kann die höhere Frequenz aber gut bestimmt werden.

Die Erkennung der niedrigeren der beiden Frequenzen ist etwas schwieriger. Nach og. Tabelle sind jeder der drei hohen Frequenzen (1209Hz, 1336Hz, 1477Hz) jeweils vier niedrige Frequenzen zugeordnet die unterschieden werden müssen.
Aus dem eingehenden Signal werden hierfür die ADC-Werte von 30 Wellenbergen und 29 Wellentälern gespeichert. Diese 59 Werte (8bit) reichen zur Bestimmung aus. Nach entsprechender Aufbereitung werden die Werte mit den im Programm gespeicherten Mittelwerten verglichen. Sie sind mit *DTMF_EXCEL_OUT.asm,* hier ebenfalls beiliegend, erstellt worden.
Bei ausreichender Übereinstimmung erfolgt die Ausgabe via 7-Segment-LED-Anzeige. Der kodierte Tastenwert wird zudem in der Variable TASTE gespeichert.

Für eigene Experimente sind hier sämtliche DTMF-Signale als mp3-Datei angehängt. Kleiner Tipp: Als Abspielgerät eignet sich ein Mobiltelefon!

Im folgenden werden die einzelnen Programmsegmente dargestellt.

Hauptprogramm

Nach den üblichen Konstanten- und Variablendefinitionen werden die beiden benötigten Interrupts (ITR's) festgelegt:
TIM0_OVF:
Der ATtiny26 besitzt nur einen 8bit-Zähler. Dieser reicht für die Frequenzanalyse nicht aus. Der 16bit-Zähler wird deswegen als Software-Lösung gebildet:
Mit jedem Overflow des 8bit-Zählers wird ein ITR ausgelöst und die Variable uSec_H_ITR (als High-Byte) inkrementiert.
ADC_COMPLETE:
Der ADC-Wandler arbeitet im schnellen *free–running-mode*. Nach jeder fertigen Messung wird ein ITR ausgelöst.
Das eigentliche Hauptprogramm ist nur eine Endlosschleife.

ADC_START

Initialisierung des ADC-Wandlers:
Das Ergebnis ist *left-adjusted* (8bit). Wegen der geringen Auflösung kann der ADC-Takt auf 250Khz erhöht werden. Als Eingang wird ein *Differential ADC Channel Pair* mit 20-facher Verstärkung verwendet.

ADC_COMPLETE

Dies ist die wichtigste Routine. Sie wird via Interrupt immer dann aufgerufen, wenn eine ADC-Messung beendet ist.
Nach dem Auslesen des 16bit-Timers [uSec_H:uSec_L] werden ITR's sofort wieder ermöglicht (*sei*), um den Timer nicht zu stören.
Die sog. „Berge" bzw. „Täler" der Schallwellen werden erkannt, wenn eine ansteigende Welle plötzlich wieder abfällt bzw. eine abfallende Welle wieder ansteigt. Es werden 30 Berge und 29 Täler im SRAM gespeichert. Ebenso wird der kleinste und größte ADC-Wert in den Variablen Min und Max gespeichert.
Sind 59 Messwerte abgezählt, wird der Zähler gestoppt und die Auswertung kann beginnnen.
Die Suche der höheren Frequenzen erfolgt in der Routine FREQ_CHECK. Die zum Vergleich zuvor eingespeicherten Werte *[F1209_H:F1209_L]*, *[F1336_H:F1336_L]* und *[F1477_H:F1477_L]* wurden mit der folgender Formel berechnet:

$$f(Messung) = \frac{30}{\mu Sec\_H \times 256 + \mu Sec\_L} \times 10^6$$

Wurde keine Frequenz gefunden, beginnt das Programm von vorne: Es werden neue ADC-Werte gespeichert usw…
War die Frequenzsuche erfolgreich, kommt es zur Analyse der unteren Frequenz.

TIM0_OVF

Wie bereits beschrieben: Kurze ITR-Routine zum Inkrementieren von uSEC_H_ITR.
Ziel: Bestimmung der Zeit (in µs), die zum Durchlaufen von 30 Wellenbergen notwendig ist.

RESTART_uSec

Reset und Initalisierungen für den Neustart

FREQ_CHECK

Für die höheren Frequenzen sind Referenzwerte für die benötigte Zeit im Programm hinterlegt. Der gemessene Wert muss innerhalb des aus Referenzwert [ZH:ZL] und Toleranzwert [Toleranz_H:Toleranz_L] gebildeten Intervalls liegen, um als eine der drei Frequenzen erkannt zu werden. Das Ergebnis wird in der Variable High_Freq gespeichert.

SIGNAL_CHECK

Routine, um die gespeicherten 59 Berg- und Talwerte für die weitere Analyse vorzubereiten:

1. Entfernung des Gleichstromanteils:
   Während der Messwertaufnahme wurde der kleinste Tal-Wert in *Min* gespeichert. Dieser Wert wird von allen Bergen und Tälern subtrahiert.

2. Werte „aufweiten“:
   Alle Berge und Täler sollen so angepasst werden, das der kleinste Wert etwa bei 1 und der grösste bei 255 liegt. Dies soll durch Multiplikation mit einem sog. *Faktor* erfolgen. Damit dieser *Faktor* nur aus acht Bit bestehen kann, werden die Werte ggf. zuvor mehrfach mit zwei multipliziert.
   Für ausreichende Genauigkeit müßte eigentlich eine Division mit mindestens zwei Nachkommastellen im Ergebnis durchgeführt werden:

   1. $Faktor = 255 / Maximalwert$
   2. $(Y_n) = (Y_n) \times Faktor \qquad (für\ n = 1 .. 59)$

   hier hilft ein Trick:

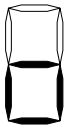
   1. $Faktor' = 25500 / Maximalwert$
   2. $(Y_n) = (Y_n) \times Faktor' \qquad (für\ n = 1 .. 59)$

3. $(Y_n) = (Y_n) / 100$

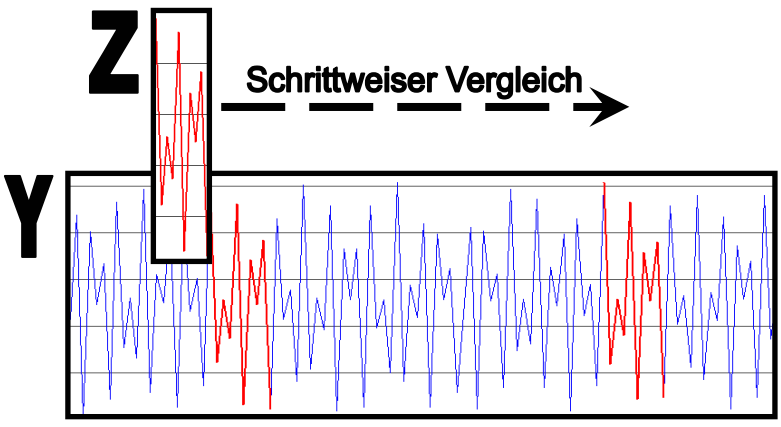
Alle Divisionen und Multiplikationen werden durch mehrfache Subtraktionen bzw. Additionen ausgeführt. Dies dauert zwar etwas länger, ist aber sehr einfach und „speicherschonend“ zu programmieren.

CHECK_TASTE

Hier erfolgt der Vergleich der modifizierten Messwerte mit den im Programmcode gespeicherten Referenzwerten für die tiefere Frequenz.
Da zuvor schon die höhere Frequenz ermittelt wurde, reicht es, die Messwertreihe mit jeweils nur vier Referenzwertreihen zu vergleichen. Die typischen Ausschnitte der Interferenzmuster, die hier gesucht werden, können an einer beliebigen Stelle beginnen. Deswegen wird die Messwertreihe gegen die Referenzwertreihe stückweise verschoben und dann immer wieder erneut verglichen. Hierbei ist eine maximale Abweichung von Konstante *Tol_Ref* nach oben oder unten erlaubt.
Zuletzt erfolgt ggf. die Ausgabe der gefundenen Taste mithilfe der 7-Segment-Anzeige.

Raute: 

Stern: 



SET_Z

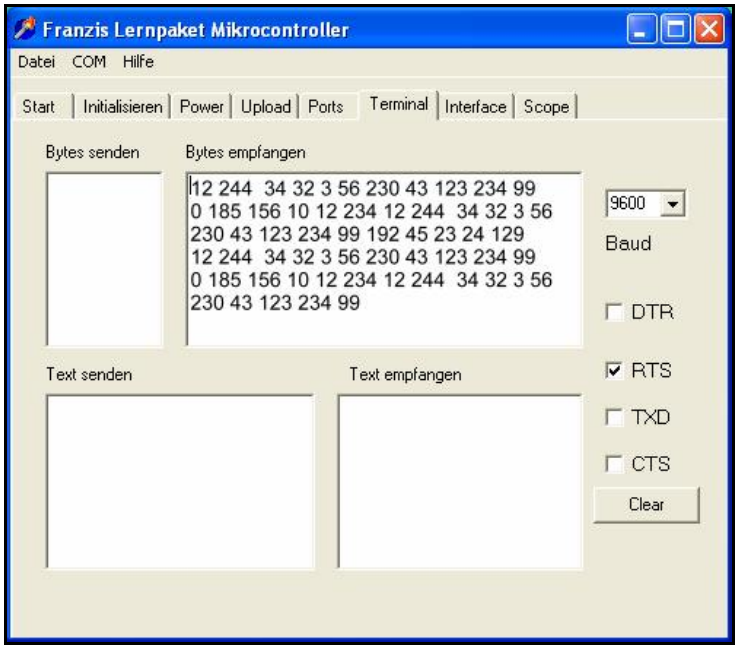
Diese Routine wird in CHECK_TASTE verwendet, um das Z-Register stets auf den ersten Wert der aktuell benötigten Referenzwertreihe einzustellen. Ausserdem wird in Variable *Taste* der Tastencode (NULL, EINS, …) gespeichert. Die Variablen *q* und *r* werden für die spätere Ausgabe in der 7-Segment-Anzeige benötigt.

DTMF_EXCEL_OUT

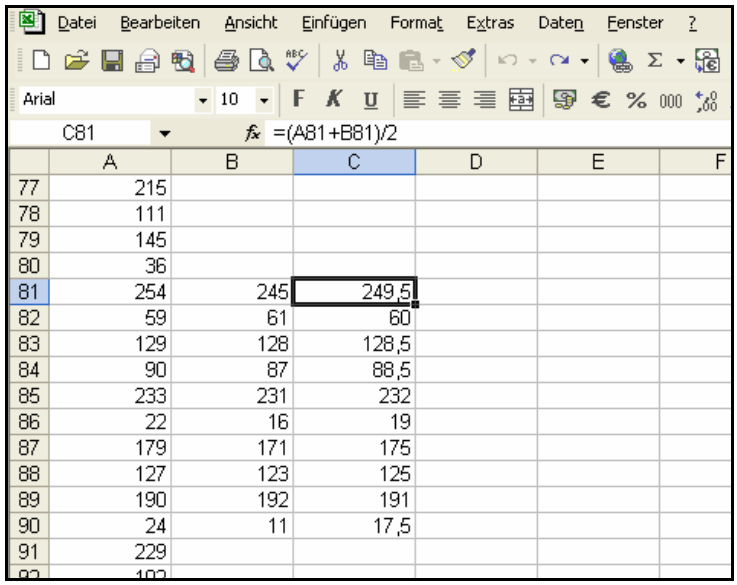
Allgemeines

Kann das Terminal von ***Lpmikros.exe*** aus dem *Lernpaket Mikrocontroller* verwendet werden, so lassen sich hier mithilfe der Routine ***WrCOM*** diverse Werte an den Computer übertragen. Eigentlich ist dieses Programm eine Vorversion des fertigen DTMF-Decoders ohne Erkennung der tieferen Frequenz.
Hiermit wurden die Referenzwertreihen ermittelt:

1. Terminal von Lpmikros.exe starten
2. DTMF-Signal über Lautsprecher vor dem Mikrofon abspielen.
3. Taste an PA5 zur Werteaufnahme zweimal drücken.
4. Die 2 x 59 Werte aus dem Terminal kopieren und für EXCEL vorbereiten.
5. Tabellen und Grafiken mit EXCEL erstellen.
6. Wiederkehrende Ausschnitte von Interferenzmustern aus den Grafiken optisch ausfindig machen.
7. Die zugehörigen Tabellenwerte selektieren und für mehrere Ausschnitte Mittelwerte errechnen lassen.

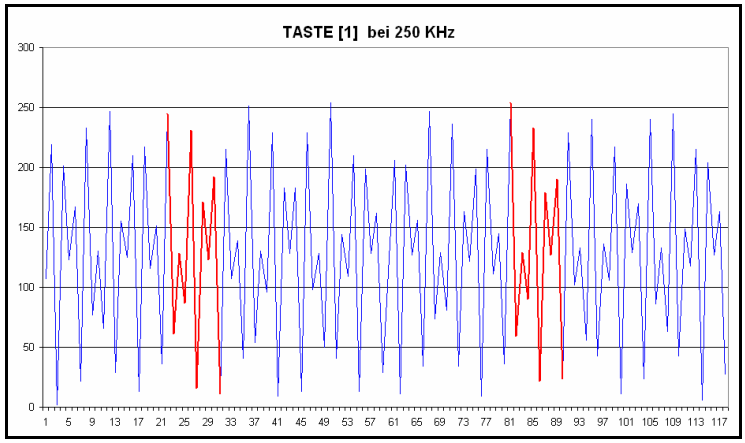


Wertaufnahme durch das Terminal mit WrCOM



	A	B	C	D	E	F
77	215					
78	111					
79	145					
80	36					
81	254	245	249,5			
82	59	61	60			
83	129	128	128,5			
84	90	87	88,5			
85	233	231	232			
86	22	16	19			
87	179	171	175			
88	127	123	125			
89	190	192	191			
90	24	11	17,5			
91	229					

Import nach EXCEL und Mittelwerte berechnen



Darstellung der wiederkehrenden Muster