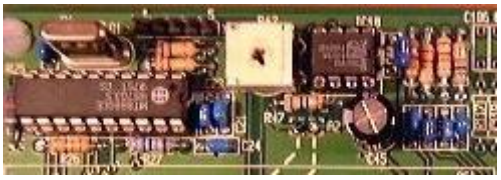
Der MT8880C DTMF - Transceiver

Ein Superchip für die Erkennung und Erzeugung von DTMF – Tönen.



Als Philip Reiss das Telefon erfand, dachte er sicherlich noch nicht daran, dass man irgendwann einmal Millionen Menschen über das Telefon weltweit verbinden kann. Für ihn war es wichtig, dass das Gespräch von A an der Stelle B überhaupt mehr schlecht als recht empfangen wurde. Graham Bell, dem man sicherlich bestätigen muß, dass er das praktisch nutzbare Telefon erfunden hat, ging es sicherlich ähnlich.

Doch schon Anfang dieses Jahrhunderts besann man sich auf eine Vermittlungstechnik, die automatisch die Teilnehmer verbinden sollte. Doch lange Zeit noch war man auf die Handvermittlung angewiesen. Der Teilnehmer A rief sein „Amt“ an, dort sass das Fräulein vom Amt und stöpselte die Leitung von A auf die Leitung von B. Das Gespräch konnte stattfinden.

Als dann die Drehwähler erfunden wurden, wurde es einfacher. Es war eine Supererfindung, die sich über Jahrzehnte bewährte. Die gewählte 9 ratterte 9mal bis zum Kontakt 9. Eine 0 ratterte 10 mal bis zum Kontakt 10 usw. Letztendlich war die Verbindung geschaffen – ohne Zutun menschlicher Hände.

Das Verfahren hat uns in Deutschland lange begleitet. IWV – das Impulswählverfahren – ist gar heute noch weit verbreitet, obgleich mittlerweile alle Vermittlungsämter auf ISDN umgestellt wurden.

In den sechziger Jahren wurde in den Bell Labs ein anderes Wählverfahren entwickelt: das Wählen über Töne, dem sogenannten DTMF-Verfahren. DTMF heißt Dual tone multi frequency. Das Wählen einer Telefonnummer wird nicht mehr über Impulse (ein/aus) realisiert, hier werden spezielle Töne gesendet, die eindeutig einer Zahl zugeordnet werden können.

Jeder DTMF – Ton besteht aus einem Frequenzpaar – die Frequenz low und die Frequenz high. Beide Frequenzen werden beim Senden über die Leitung gleichzeitig gesendet. Ein Beispiel: die Ziffer 1 wird mit einer gemischten Frequenz aus 697 Hz (Flow) und 1209 Hz (Fhigh) ausgestrahlt, eine 10 = 0 auf der Telefontastatur mit 852 Hz (low) und 1336 (high) dargestellt.

F_{LOW}	F_{HIGH}	DIGIT	D_3	D_2	D_1	D_0
697	1209	1	0	0	0	1
697	1336	2	0	0	1	0
697	1477	3	0	0	1	1
770	1209	4	0	1	0	0
770	1336	5	0	1	0	1
770	1477	6	0	1	1	0
852	1209	7	0	1	1	1
852	1336	8	1	0	0	0
852	1477	9	1	0	0	1
941	1336	0	1	0	1	0
941	1209	*	1	0	1	1
941	1477	#	1	1	0	0
697	1633	A	1	1	0	1
770	1633	B	1	1	1	0
852	1633	C	1	1	1	1
941	1633	D	0	0	0	0

Die verschiedenen Frequenzen und das zugehörige Bitmuster

Die Wissenschaftler, die die DTMF – Telefonwähltechnik ersannen, haben lange nach den richtigen Frequenzen gesucht. Das Ziel war: es mußten Frequenzpaare gesucht werden, die in der menschlichen Sprache niemals vorkommen können. Es wäre nämlich fatal, wenn man während des Gespräches plötzlich Schaltvorgänge mit der menschlichen Stimme auslösen könnte.

Die Auswahl der Frequenzen hat sich bewährt, denn nach und nach nach werden alle Telefonsysteme auf DTMF umgestellt. Der Vorteil liegt auf der Hand: eine Wahl mit DTMF geht bedeutend schneller, als mit dem überholten Impulswahlsystem.

Man kann sich dies einfach verdeutlichen: Beim Impulswahlverfahren ist die Geschwindigkeit des Wählvorganges abhängig von den zu wählenden Ziffern. Eine 0 sind 10 Impulse, eine 2 sind nur 2 Impulse. Jeder Impuls dauert. Eine Telefonnummer 099999 (gibt es zwar nicht) hätte 10 + 45 = 55 Impulse.

DTMF löst das Problem besser. Jede Ziffer benötigt zur Übermittlung die gleiche Zeit, da sie nur aus einem Frequenzpaar dargestellt wird. Damit ist es gleichgültig, ob die Ziffer groß oder klein ist.

Das Tastverhältnis von Tönen und Pausen ist knapp bemessen. In Amerika kann man davon ausgehen, dass die Übermittlung von 51 Millisekunden Tonübertragung und 51 Millisekunden Pause zum Erfolg führt.

Bei uns sollte man etwas langsamer senden, weil nicht alle Anlagen dieses Format verstehen. Um es vorwegzunehmen: Der Chip MT8880 besitzt die Möglichkeit der Verdoppelung der Zeiten: 102 Millisekunden Tonausgabe, 102 Millisekunden Pause. Damit sind wir immer noch schneller als beim Impulswahlsystem. Bei einer zehnstelligen Nummer liegen wir bei knapp 2 Sekunden.

Zur Übermittlung von Steuersignalen (nach einer aufgebauten Verbindung) eignen sich nur die DTMF-Töne, da sie selbst bei schlechten Verbindungen meistens noch sicher erkannt

werden können. Versuche, die wir früher mit dem Impulswählverfahren unternahmen, waren alle erfolglos, weil keine gesicherte Datenübertragung realisiert werden konnte.

Bei der ersten Version von Lallus war ein MT8870 im Einsatz, der die gesendeten Steuersignale empfing, sie dekodierte und zu Steuerungen führte. Das klappte schon wunderbar. Jetzt aber wollen wir den Transceiverchip MT8880 besprechen, der zweierlei bewerkstelligen kann: Empfang von DTMF - Signalen und Senden von DTMF - Signalen in das Telefonnetz.

Dieses hat den Vorteil, dass nicht nur Signale entgegengenommen werden können – bei Bedarf kann auch über diesen Chip eine Telefonnummer gewählt werden.

Der MT8880 ist nicht ganz so trivial zu programmieren wie der 8870, der ja nur eine Richtung kennt.

Außerdem „frisst“ er 8 Digitalports, um die nötigen Ansteuerungen vornehmen zu können. Bei der Entwicklung von Lallus 2 hat dies zunächst einmal „sehr weh getan“.

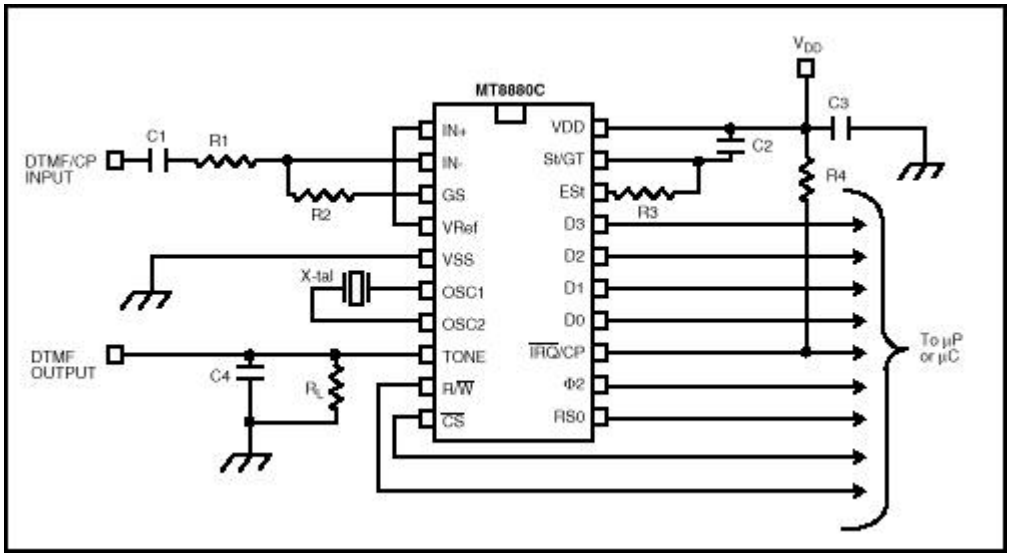
Ein ganzer Byteport mit 8 Ein/Ausgängen mußte geopfert werden. Letztendlich aber ist es wichtig, dass Lallus auch nach außen hin agieren kann.

Die Alternative wäre folgende gewesen: das Senden der DTMF-Töne hätte man über den Soundchip ISD2560 realisieren können. Die DTMF-Tonsegmente hätte man dort bereithalten können. Die Beobachtung der Leitung (besetzt, frei, angenommen) hätte man evtl. über den Frequenzeingang der CControl lösen können.

Empfang von DTMF – Signalen.

Der MT8880 kann dies eleganter.

Hinweis: Sollten Ihre Telefone noch auf Impulswahl stehen, so sollte man versuchen, diese auf das bessere MFW (Multifrequenzwahl) umzustellen. Wenn dies nicht geht, müssen Sie Ihre Versuche mit einem Tongeber (Anrufbeantworter – Fernabfrage) machen. An dieser Stelle sei auch erwähnt, dass es natürlich günstig ist, wenn man über eine Telefonanlage mit mindestens zwei Leitungen verfügt. Es könnte teuer werden, wenn man alle Versuche mit „echten“ Telefonaten unternehmen muss.



Der MT8880C mit seinen Anschlüssen.

Die einzelnen Funktionen werden wir später noch kennenlernen. Wir erkennen auch, dass eigentlich 9 Ports eines Mikrocontrollers belegt werden, wenn man sich die rechte Seite mit ihrer yP – Klammer ansieht. Der CS – Port (Chip Select) wurde bei unserer Lösung weggelassen, da man typischerweise nur einen MT8880 einsetzt.

Der Nachteil ist darin zu sehen, dass man jetzt den Chip nicht mehr abschalten kann. Ein paar Milliamperes nehmen wir für den gesparten Port in Kauf.

Von den 16 Digitalports müssen wir gar 9 zur Bedienung der Telefonfunktionen opfern. Zusätzlich wurde noch ein Analog/Digitalport reserviert. Dieser AD-Port (AD 8) entdeckt einen Anruf. Der 9. Digiport bedient das Auflegen oder Abheben des Telefons. Damit ist schon ein großer Teil der vorhandenen Ports für das Telefon reserviert.

Zum Glück gibt es den I2C-Bus, der die fehlenden Ports später ersetzen kann.

Der Deklarationsteil zeigt dies:

```
define klingel   ad  [8]
define abgehoben port[5]
define b3        port[9]
define b2        port[10]
define b1        port[11]
define b0        port[12]
define irq       port[13]
define phi2      port[14]
define rs0       port[15]
define rw        port[16]
```

‚abgehoben‘ bezieht sich auf die Telefonfunktion: Telefon auflegen oder abnehmen – das, was wir sonst manuell ausführen.

‚klingel‘ ist eigentlich auch ein digitaler Wert, der zwischen 0 und 1 wechselt. Da die Digitalports knapp wurden, ist diese Funktion auf einen Analogport ausgelagert worden.

Das Lineinterface liefert eine logische Null, wenn ein Klingelsignal auf der Telefonleitung anliegt, ansonsten steht hier eine 1 mit dem Wert 255 an. Der Analogport ‚*klingel*‘ wird nach 0 gezogen. Eine Abfrage kann jetzt gestartet werden: `if klingel < 100 then` .....

Ist `klingel < 100`, dann hat sich auf der Leitung etwas getan und das Programm kann darauf reagieren. Man kann jetzt einen Zaehler laufen lassen. Wenn das Klingelsignal z.B. 3 mal nach unten gezogen wurde, so kann der Befehl `abgehoben=on` ausgeführt werden. Damit ist die Verbindung hergestellt.

Jetzt heißt es: die nachfolgenden DTMF-Töne müssen erkannt werden, um evtl. Schaltvorgänge auszuführen.

Der Programmcode gibt Auskunft:

```
#warten
if klingel >100 then goto warten
print "Ruf angenommen - warte auf DTMF"
print "Modus beenden mit # auf Telefon"
abgehoben = on
```

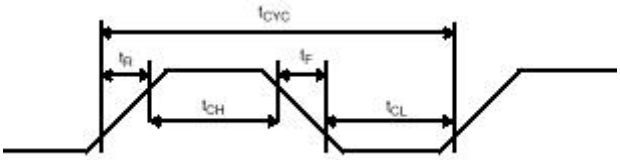
```
phi2 = on
rs0=1:rw=1
b3=0:b2=0:b1=0:b0=0
deact b3:deact b2:deact b1:deact b0

#status
rs0=1 : rw=1 : pulse phi2
if b2=0 then goto status
rs0=0 : rw=1 : pulse phi2
zeichen=b3*-8 + b2*-4 + b1*-2 + b0*-1
print "empfangener Wert: ";zeichen
if zeichen=12 then goto anzeige
goto status
```

Interessant ist das Unterprogramm ‚status'. Mit `rs0=1 : rw=1 : pulse phi2` wird ein Registeraufruf vorgenommen.

Der Chip liefert jetzt an der Stelle von Bit 2 eine Information, ob ein neu eingegangener gültiger DTMF-Ton vorliegt. Solange b2=0 ist, hat sich hier nichts getan. Ist b2 <> 0, dann liegt ein neuer erkannter Wert vor.

Jetzt kann man den Aufruf `rs0=0 : rw=1 : pulse phi2` starten. Dieses bedeutet: lese das Empfangsregister aus.

Phi2 ist das Clocksignal, das man jeweils für die Mitteilungen an den Chip aufrufen muß.



An dem Bild erkennen wir, dass der Chip auf die negative Triggerflanke von phi2 reagiert.

Hinweis: Beim Schreiben müssen wir phi2=off, beim Lesen phi2=on berücksichtigen.

Nach dem Taktimpuls phi2 können die einzelnen Bits jetzt zusammengefügt werden, da der Chip die Ausgänge b3 - b0 mit den entsprechenden Werten darstellt.

Wir könnten jetzt die einzelnen Ausgänge mit IF-Abfragen zu einem Wert zusammenzählen.

```
If b3=1 then wert=wert+8
If b2=1 then wert=wert+4
If b1=1 then wert=wert+2
If b0=1 then wert=wert+1
```

Die schnellste, platzsparendeste und eleganteste Methode zur Ermittlung stellt untenstehende Methode dar:

```
Zeichen = b3*-8 + b2*-4 + b1*-2 + b0*-1
```

Ist der Digitalport high, so liefert er eine –1, ist er logisch 0, so lesen wir eine 0.

-1 * – 8 ergibt nach Adam Riese: 8.
0 * – 8 entsprechend 0.

Alle Bits werden mit ihrer Wertigkeit addiert. Das Ergebnis wird in ‚*zeichen*' gespeichert.

Im obigen Programmbeispiel entdecken wir eine Zeile, die vielleicht nicht sofort klar ist:

```
deact b3:deact b2:deact b1:deact b0
```

Mit dem Befehl ‚*deact*' werden die Digitalports auf Eingang gestellt. Evtl. Potenzialwechsel an den Eingängen werden nunmehr erkannt. Wenn der 8880 eine ‚1' vorlegt, so wird dies in diesem Eingangsmodus verstanden. Steht dagegen ‚1' als Ausgang, so behält er sein Potenzial.

Mit dem obigen Programmteil können wir die Steuersignale, die von aussen kommen detektieren. Wie sieht es jetzt aus, wenn Lallus selbst aktiv wird?

Wählen mit DTMF

Stellen wir uns folgendes vor: eine Fensterscheibe, die durch einen Sensor am System angeschlossen ist, meldet Alarm. Jetzt ist die Schaltung gefragt, eine Aktion über Telefon auszuführen.

Der MT8880C kann dies bestens erledigen. Die Telefonnummer, die in einem solchen Fall gewählt werden soll, sei im Speicher des EEPROMS 24C65 abgespeichert. Sie kann natürlich auch in einem TABLE gespeichert sein.

Das TABLE wird man benutzen, wenn man die Priorität ‚handy' anrufen will, da diese Nummer sich selten ändert.

Das EEPROM kann man benutzen, wenn man von außerhalb Nummern zum Speichern eingeben will. Also: zunächst einmal das Handy anrufen; wenn keine Antwort – die gespeicherte Nummer. Oder: Das Handy ist nicht erreichbar, da man sich an einem Ort befindet, der keinen Kontakt zum Funknetz hat – man kann eine Festnetznummer speichern.

Um alles realisieren zu können, müssen wir eine Menge lernen. Letztendlich ist aber alles möglich.

Fangen wir zunächst einmal mit TABLE – Aufrufen an. Das, was man in TABLE Name und TABEND hineinschreibt, kann man jederzeit wieder abrufen.

```
TABLE Telefon 1 2 3 4 5 6 7
TABEND
```

Der Aufruf *table telefon , 0 , wer*t füllt die Variable ‚wert' mit 0, da der erste Indexwert der Tabelle abgefragt wurde. So kann man die Handynummer eintragen:

```
Table handy 0 1 7 2 1 2 3 4 5 6 7 8 9 99
Tabend
```

Das Ende der Nummer kann man mit einer Zahl, die sonst nicht vorkommt, markieren. Im obigen Beispiel ist es die 99. Sobald man diese Zahl liest, ist die Telefonnummer zu Ende und man kann verzweigen.

Wie angesprochen, so kann man die zu wählende Nummer auch im EEPROM speichern. Sofern der Speicher nicht vollständig aufgebraucht ist, kann man hier Informationen ablegen, die man ohne Stromversorgung (bis zu 10 Jahre) im Speicher belassen kann.
Die Befehle dazu sind recht einfach zu handhaben:

```
Open# for write
Close#
```

Die Datei wird für den Schreibmodus geöffnet.
Alles, was vorher gespeichert war, ist jetzt verloren, da ein *close#* direkt hinterherkam.

Der Appendmodus ist für unser Ziel hier sehr geeignet.

```
Open# for append
Print# 11
Close#
```

An die erste Stelle des feien Speichers wurde eine 11 geschrieben.
Ein weiterer Aufruf

```
Open# for append
Print# 15
Close#
```

Schreibt eine 15 in den Speicher. Jetzt ist 11 und 15 gespeichert, Dieses kann man sich anzeigen lassen.

```
Define wert byte
Open# for read
#nochmals
Input# wert
If eof then goto ende
Print wert
Goto nochmals
#ende
close#
Print „Datei gelesen"
end
```

Kommen wir zurück zum Chip MT8880C. Es soll ein DTMF-Ton erzeugt werden. Um dieses ausprobieren zu können, müssen natürlich vorab die einzelnen Ports definiert werden.

```
define b3   port[9]
define b2   port[10]
define b1   port[11]
define b0   port[12]
define irq  port[13]
define phi2 port[14]
define rs0  port[15]
define rw   port[16]
```

phi2 ist der Takteingang, der oben schon angedeutet wurde. Das `pulse` – Signal, das unten benutzt wird, bringt den phi2-Port kurzzeitig auf high und fällt dann wieder auf `low`. Beim Wechsel von high nach low reagiert der Chip und übernimmt die angelegten Werte. Beim Senden wählen wir: `phi2=off`

RS0	R/$\overline{W}$	FUNCTION
0	0	Write to Transmit Data Register
0	1	Read from Receive Data Register
1	0	Write to Control Register
1	1	Read from Status Register

Dem Datenblatt können wir entnehmen, dass der Chip über vier Steuerregister verfügt, je nach Belegung der Eingänge rs0 und rw.

Soll ein Wert gesendet werden, so benutzen wir das Transmitregister `rs0=0:rw=0`.
Wollen wir wissen, was von aussen gesendet wurde, so Lesen wir das Empfangsregister mit `rs0=0:rw=1`, was oben bereits benutzt wurde.

Wollen wir verschiedene Parameter dem Chip übergeben, so wird das Controlregister benutzt **`rs0=1:rw=0.`** Davon gibt es 2; Controlregister A und Controlregister B. Wir sehen das gleich an unserem Beispiel.

Bleibt noch das Statusregister **`rs0=1:rw=1`**. Dieses Register benutzt man, um die Zustände genau abfragen zu können. Z.B.: Ist ein neuer gültiger DTMF-Ton eingegangen?

```
rs0=1 : rw=0 : b3=1 : b2=0 : b1=1 : b0=1 : pulse phi2
rs0=1 : rw=0 : b3=0 : b2=0 : b1=0 : b0=0 : pulse phi2
```

Die obigen Programmzeilen sind sicherlich erklärungsbedürftig. `rs0=1 : rw=0` entspechen der Registerwahl: Schreibe in das Controlregister. Doch was?

Dem Datenblatt kann man die Bedeutungen entnehmen. Wir sehen oben zwei ähnliche Programmzeilen.

Dass es zwei sind bedeutet: wir benutzen Controlregister A und Controlregister B. Der Chip erfährt dies mit der `b3=1` (REGISTER SELECT) in der ersten Zeile. Wir signalisieren: Achtung es kommt ein zweiter Schreibvorgang. Stünde hier `b3=0`, so wäre nur das Controlregister A betroffen.

Das nächste Bit wird auf 0 (INTERRUPT ENABLE) gesetzt. Es kann später benutzt werden, um einen Interrupt zu erzeugen. Dieser Interrupt tritt auf, wenn auf der Leitung etwas passiert und kann abgefragt werden.

Weiter mit der ersten Zeile, nun `b1=1` . CP/DTMF ist dieses benannt – ein wichtiges Bit bei der Steuerung des Chips.

Jetzt müssen wir schon etwas ausholen. Eine logische 0 bedeutet, dass u.a. DTMF – Töne gesendet und empfangen werden können. Das wollen wir ja erreichen, doch warum steht hier eine 1? Senden kann man auch mit einer logischen 1, empfangen nur mit einer 0. Die

zusätzliche Bedeutung: ist die 1 gewählt in Verbindung mit dem Burstsignal (b0 der zweiten Zeile), so wird die Tondauer des gesendeten Signals statt 51 ms auf 102 ms und die anschließende Pause ebenfalls auf diesen Wert gesetzt. Warum dieses günstig ist, habe ich oben bereits beschrieben.

Das nächste Bit der ersten Zeile: `b0=1` (TONE OUTPUT). Dieses ist ganz wichtig, denn steht hier eine 0, so wird der Ton zwar intern erzeugt, jedoch nicht zur Ausgabe gegeben.

Es folgt der Impulsbefehl `pulse phi2`, der die oben eingestellten Werte dem Chip übermittelt. Controlregister A wäre damit bedient. In Bit3 stand aber eine 1, so dass ein zweiter Registereintrag für das B-Register erwartet wird. Wir betrachten die zweite Zeile.

```
rs0=1 : rw=0 : b3=1 : b2=0 : b1=1 : b0=1 : pulse phi2
rs0=1 : rw=0 : b3=0 : b2=0 : b1=0 : b0=0 : pulse phi2
```

`rs0=1:rw=0` ist bekanntlich: Schreiben in das Controlregister – wie gehabt.

`b3 = 0` (COLUMN/ROW TONES).Hier wählt man die Art der Tonausgabe. Für Testzwecke oder andere Einsätze kann man auch eine einzige Frequenz ausgeben. Entweder eine high – Frequenz oder eine Low-Frequenz. Für uns interessiert hier aber das DTMF-Signal mit seiner Mischfrequenz – also 0.

`b2 = 0` (SINGLE/DUAL TONE GENERATION). Ähnlich wie oben bereits beschrieben. DTMF oder Einzelfrequenz – hier also 0.

`b0 = 0` (BURST MODE) bedeutet, dass dieser Modus eingeschaltet ist. Jetzt kann ein Tonpaar mit der gwählten Länge (51 oder 102 ms) ausgegeben werden. Wenn hier eine 1 stehen würde, so hat dies auch eine Bedeutung, die wir hier nicht erörtern wollen.

Damit sind die Controlregister beschrieben, wenn `pulse phi2` ausgeführt ist. Zusammengefasst haben wir mitgeteilt: Ein Ton von 102 ms Länge und 102 ms Pause soll ausgegeben werden. Welcher Ton? Das kommt jetzt in einem weiteren Schritt.
Zunächst legen wir das gewünschte Bitmuster an die Dateneingänge b3, b2, b1, b0. Entsprechend der Wertigkeit teilen wir eine Ziffer auf. Zunächst eine 8. `b3=1:b2=0:b1=0:b0=0.`

Ein zweites Beispiel mit einer 7: `b3=0:b2=1:b1=1:b0=1.`

Natürlich wäre es Unsinn, wenn wir immer per Hand die Bitfolge eingeben müssten. Dieses macht natürlich eine kleine Routine:

```
Define zeichen byte

Zeichen=(1-16)

 b3=zeichen and 8  oder auch: b3=zeichen and 1 shl 3
 b2=zeichen and 4  oder auch: b2=zeichen and 1 shl 2
 b1=zeichen and 2  oder auch: b1=zeichen and 1 shl 1
 b0=zeichen and 1  oder auch: b0=zeichen and 1 shl 0
```

Nunmehr liegen die gewünschten Bitwerte an den Eingängen. Diese Werte müssen jetzt vom Chip übernommen werden. Es ist sicher klar, dass wir jetzt das Transmitregister benutzen müssen, um die Töne auszugeben. Der Impuls löst den Ton dann aus.

```
rs0=0:rw=0:pulse phi2
```

Noch einmal das komplette Programm zur Erzeugung eines Tones. Kopieren Sie das Programm in den CControleditor und probieren Sie es einmal aus. Sie können dann auch die Werte der Controlregister ändern, um etwas Praxis mit dem Chip zu bekommen. Die serielle Schnittstelle wird überwacht, so dass Sie durch einfache Tastatureingaben die verschiedenen Töne erzeugen können.

Die Werte 10,11,12,13,14,15 und 16 können Sie mit CTRL (STR) und J (10) , K (11) usw. gleichzeitig gedrückt senden.

```
define b3 port[9]:define b2 port[10]:define b1 port[11]:define b0
port[12]
define irq port[13]:define phi2 port[14]:define rs0 port[15]:define
rw port[16]
define zeichen byte:phi2=off
#main
zeichen=99
if rxd then get zeichen
print zeichen
if zeichen=99 then goto main
rs0=1 : rw=0 : b3=1 : b2=0 : b1=1 : b0=1 : pulse phi2
rs0=1 : rw=0 : b3=0 : b2=0 : b1=0 : b0=0 : pulse phi2
b3=zeichen and 8:b2=zeichen and 4: b1=zeichen and 2:b0=zeichen and 1
print b3,b2,b1,b0
rs0=0:rw=0:pulse phi2
goto main
```

Um den Umgang mit Dateien zu verdeutlichen, habe ich hier ein weiteres kleines Porgramm angefügt, das die zu sendenden DTMF – Töne aus dem EEPROM holt. Es ist sicher leicht zu verstehen.

```
define b3 port[9]:define b2 port[10]:define b1 port[11]:define b0
port[12]
define irq port[13]:define phi2 port[14]:define rs0 port[15]:define
rw port[16]
define zeichen byte:phi2=off

open# for write
for zeichen=1 to 16
print# zeichen
next zeichen
close#

open# for read
#lesen
if eof then goto ende
input# zeichen
rs0=1 : rw=0 : b3=1 : b2=0 : b1=1 : b0=1 : pulse phi2
rs0=1 : rw=0 : b3=0 : b2=0 : b1=0 : b0=0 : pulse phi2
b3=zeichen and 8:b2=zeichen and 4: b1=zeichen and 2:b0=zeichen and 1
print b3,b2,b1,b0
rs0=0:rw=0:pulse phi2
pause 10
goto lesen
#ende
end
```

Auch das Arbeiten mit Tabellen soll hier kurz realisiert werden, um die Erfahrung mit der CControl zu erleichtern.

```
define b3 port[9]:define b2 port[10]:define b1 port[11]:define b0
port[12]
define irq port[13]:define phi2 port[14]:define rs0 port[15]:define
rw port[16]
define zeichen byte:define i byte :phi2=off

#nochmals
looktab telefon, i, zeichen
if zeichen=99 then end
rs0=1 : rw=0 : b3=1 : b2=0 : b1=1 : b0=1 : pulse phi2
rs0=1 : rw=0 : b3=0 : b2=0 : b1=0 : b0=0 : pulse phi2
b3=zeichen and 8:b2=zeichen and 4: b1=zeichen and 2:b0=zeichen and 1
print b3,b2,b1,b0
rs0=0:rw=0:pulse phi2
pause 10
i=i+1
goto nochmals
end
TABLE telefon 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 99
tabend
```

Soweit dürfte der MT8880C jetzt beherrschbar sein. Es muß noch eine Spezialität bearbeitet werden, die ein wenig Experimentierfreude voraussetzt. Der sogenannte Callprogress. Was bedeutet dies?

Wenn wir erfolgreich eine Telefonnummer über die DTMF-Töne abgesandt haben, so wird auf der anderen Seite das Telefon klingeln und klingeln. Wann aber hat der Angerufene abgehoben oder ist die Leitung etwa besetzt, ist sie frei?

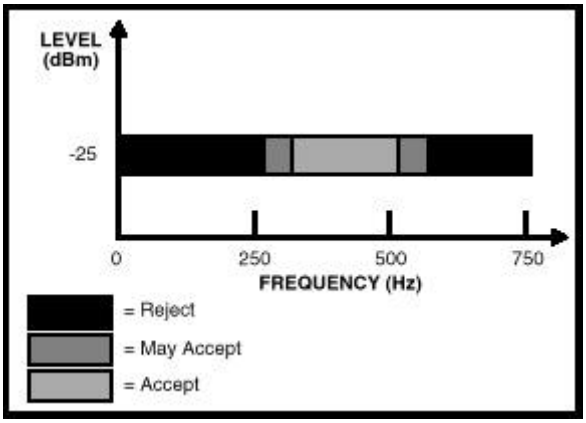
Über das Lineinterface erhalten wir keine Auskunft, der Schleifenstrom bleibt konstant und ansonsten gibt es keine verwertbare Information. Wir müssen die Leitung beobachten. Das Freizeichen tüt, pause, tüt muß ausgewertet werden. Das tüt ist nichts anderes als ein Ton, der bei 440 Hz liegt. Kein tüt, die Pause, ist der zweite Zustand, der ausgewertet werden muß.

Die Telefontechnik definiert hier unsere Werte:

1 sec tüt – 2 sec Pause = Freizeichen
½ sec tüt – ½ sec Pause = besetzt
kein tüt = abgehoben

Mit einem Mikroprozessor könnten wir die verschiedenen Zustände natürlich detektieren (wenn er schnell genug ist). 440 Hz ist natürlich eine Frequenz, die auch die CControl mitbekommt. Der Frequenzeingang kann immerhin ca. 6 kHz locker darstellen. (Es gibt übrigens noch einen zweiten Frequenzeingang, der bei der CControl leider nicht direkt ansprechbar ist. Wer ein bißchen basteln will, der muß hier die Platine auftrennen und kann dann Frequenzen bis 32 kHz detektieren. Schade eigentlich, - bei der M-Control ist er herausgeführt -.)

Vergessen wir aber an dieser Stelle die CControl. Im Chip MT8880C ist nämlich ein Modus implementiert, der genau für unseren Zweck genutzt werden kann: der Callprogress-Modus.



Schaltet man diesen Modus ein, so wird ein raffiniertes Filternetzwerk innerhalb des Chips angesprochen. Sind Frequenzen auf der Leitung, so werden diese gefiltert. Sind sie kleiner als ca. 250 Hz oder größer als ca. 550 Hz, so werden sie verworfen. Es existiert noch ein ‚schmaler may accept – Bereich. Doch die wichtigen Frequenzen um die 440 Hz werden genau herausgefiltert. (Dieses hätten wir bei der Frequenzmessung mit der CControl nicht, da auch Sprachsignale in diesem Bereich liegen könnten.)

Wird also ein Klingelzeichen oder als tüt bezeichnet, auf der Leitung erkannt, so wird dies am Chip durchgeschaltet. Wir hätten jetzt am Ausgang irq des Chips ein Rechtecksignal von 440 Hz anliegen.
Eigentlich ist es schade, dass man bei der Entwicklung des Chips nicht darauf geachtet hat, dass bei Vorhandensein der Frequenz ein Digitalsignal von 1 geliefert wird, bei Nichtvorhandensein eine 0 entsprechend.

Bei der Realisierung von Lallus haben wir dies allerdings nachträglich realisiert. Ein Monoflop 74HCT123 wird jeweils getriggert, wenn dieses Rechtecksignal auftritt. Letztendlich kann man dadurch eine 1 oder eine 0 ablesen. 440 Hz entdeckt: logisch 1; 440 Hz nicht entdeckt: logisch 0.

Jetzt kann man programmtechnisch mit den Signalen etwas anfangen – das Zeitverhalten muß überprüft werden.

Dabei ist es am einfachsten, wenn man die Pausen betrachtet. Ein Zaehler wird gestartet und das Ergebnis kann dann verwertet werden.

Ist der Ton 440 Hz vorhanden, so wird der Zaehler immer auf 0 gesetzt.

Der Zaehler läuft in einer Schleife hoch, bis er wieder auf 0 gesetzt wird.

Aus dem Zaehlerergebnis kann man nun entscheiden:

Zaehler nur geringen Wert = besetzt.
Zaehler größeren Wert = Freizeichen
Zaehler sehr großen Wert = Angenommen.

Dieses ist eigentlich für einen Digitaltechniker recht unbefriedigend. Läßt man sich die Zaehlerstaende über einen Printbefehl im Terminalprogramm anzeigen, so kann man ziemlich genau die entsprechenden Zaehlerstaende erkennen. Löscht man dann den Printbefehl, so stellen sich ganz andere Zaehlerstaende ein, weil der Programmablauf durch das Print nicht gestört wurde. Hier muß man also experimentieren.

Damit ist das Problem ersichtlich: Um ein angenommenes Gespräch zu detektieren, muß man warten, bis der Zaehler den Freizeichenwert überschritten hat. In der Praxis kann dies

bedeuten, dass man bei Annahme des Gespräches etwas länger warten muß, bevor sich die Elektronik meldet. Da es sich hier aber im 2-Sekundenbereich abspielt, ist es sicherlich akzeptabel.

Wie sieht eine Callprogess – Routine in der CControlpraxis aus? Der Anruf hat bereits stattgefunden. Danach wird in die Routine `#leitung_beobachten` verzweigt. CP/DTMF wird auf 1 gestellt, um die Funktion CP (Callprogess) zu initiieren.

```
#leitung_beobachten
print "Callprogress"

phi2=off
' Register A schreiben für call progress
rs0=1 : rw=0 : b3=0 : b2 = 0 : b1= 1 : b0 = 1 : pulse phi2

#prozess
pause 4
if not irq then nullwert=nullwert+1
if irq and nullwert >10 and nullwert<30 then print "frei"
if irq and nullwert>2 and nullwert<10 then print "besetzt"
if irq then nullwert=0
if nullwert >31 then print "angenommen"
if nullwert>254 then nullwert=0

goto prozess
```

Bei dieser Funktion wurde noch kein Entscheidungsalgorithmus eingebaut, da sich jeder Benutzer hier anders verahalten wird. Was soll passieren, wenn „Besetztzeichen“ festgestellt wurde, wielange soll man das Freizeichen klingeln lassen?, was wird an Information geschickt, wenn man feststellt, dass der Ruf angenommen wurde?

Zum Schluß soll noch einmal erwähnt werden, dass mit dem Chip MT8880C ein Superchip zur Verfügung steht, der seine Arbeit bestens erledigen kann. Wollte man all seine Funktionen in eigener Hardware realisieren, so hätte man wahrscheinlich einen Tisch voller Chips. Alleine das exakte Verhalten bei der Erkennung des DTMF – Tongemisches würde einen Wust an Filteraufbauten nach sich ziehen.

Hier kann man wirklich sagen: die 12 Mark lohnen sich.