

# embedded

Rafael Deliano V1.0 ( Papier ) : 30. Juni 00  
Steinbergstr.37 V2.0 ( pdf ) : 24. Feb. 01  
82110 Germering V2.1 ( pdf ) : 29. April 04  
Tel 089/8418317 V2.1 ( pdf ) : 14. Jan 07

[j\\_r\\_d@t-online.de](mailto:j_r_d@t-online.de)

5



## Voransichts- Version

Für Bezug des Originals  
siehe FAQ auf  
[www.embeddedFORTH.de](http://www.embeddedFORTH.de)

- 1 Inhaltsverzeichnis
- Impressum
- 2 FEC: Zyklische
- 5 Scrambler
- 6 PN-Sequenz
- 10 DTMF im Überblick
- 12 DTMF in Software
- 13 DTMF testen
- 14 PIC Assembler
- 17 PIC Programmier
- 18 V24 Schaltungen

konzentriert sich auf

g in zyklische  
eregistern was  
en anderen darauf  
ngen führt.

F-Sender ergän-  
zende Beitrag über DTMF-Empfänger  
in Software muß noch eine Weile  
warten, weil das mehr im Bereich  
DSPs liegt. Darüber sollen noch  
Beiträge kommen, aber es ist sinnvoll  
mit grundlegenden Dingen wie IIR  
und FIR-Filter anzufangen.

Die Entwicklungssoftware für  
PICs schließlich zeigt am primitivsten  
Prozessor wie Postfix-Assembler  
aufgebaut sind. Wie vieles in FORTH  
bestimmt die simple innere Funktion  
das für manche ungewohnte äußere  
Erscheinungsbild.

Die Listings sind in  
nanoFORTH geschrieben. Für die  
Konvertierung in andere FORTH-  
Varianten vgl. nanoFORTH-Manual  
GP32.

# Zyklische Codes

Wenn Daten seriell übertragen werden, ist die Ausführung von Coder und Decoder mit Schieberegistern oft aufwandsärmer als über Tabellen („Standard Array“) oder Logiknetze wie in der letzten Ausgabe dargestellt. Deshalb verwenden viele Anwendungen zyklische Codes. Meist handelt es sich dabei um zyklische Hamming-Codes die eine Untergruppe der BCH-Codes sind.

$$b(x) = x^4 + 0x^3 + 1x^2 + 1x^1 + 1x^0$$

1
0
1
1
1  
MSB



LSB

Bild 1: Polynom auf Bits

## Arithmetik

Etwas Theorie kann als nötige Voraussetzung hier nicht völlig vermieden werden. Da Reed-Solomon Codes einstweilen vernachlässigt werden sollen, wird nur mit binären Datenworten, hier „Polynome“ genannt, gearbeitet, die meist krumme Länge, also nicht 8 Bit oder ein Vielfaches davon haben. In Bild 1 sei nochmal am Beispiel die Umwandlung zwischen Polynomschreibweise und binärem Datenwort dargestellt.

Es gibt auch für Schieberegister Operationen mit dem Namen Addition, Subtraktion, Multiplikation und Division. Doch funktionieren sie anders als gewohnt, nämlich streng bitweise ohne Carry. Wie man in Bild 2 sieht, entsprechen sich damit Addition und Subtraktion. Der XOR-Befehl der CPU kann diese Funktionen für Datenworte direkt ausführen.

Etwas mühsamer ist Multiplikation, am Beispiel in Bild 3 dargestellt. Erinnert aber nicht ungefähr an Shift&Add Multiplikation in der normalen Arithmetik. Das Wort „a“ wird schrittweise nach links geschoben, was die CPU mit LSL, leicht ausführen kann. Gleichzeitig erfolgt Addition durch XOR-Befehl, falls in Wort „b“ dieses Bit gesetzt ist.

Bild 2:

Addition	und	Subtraktion	$\begin{array}{r} 1\ 1\ 0\ 0 \\ +1\ +0\ +1\ +0 \\ \hline 0\ 1\ 1\ 0 \\ \text{(Carry)} \\ 1\ 1\ 0\ 0 \\ -1\ -0\ -1\ -0 \\ \hline 0\ 1\ 1\ 0 \\ \text{(Borrow)} \end{array}$
----------	-----	-------------	--

Bild 4 zeigt die Division. Den Divisor aus der Ruhelage nach links schieben, bis sein oberstes Bit das oberste Bit des Dividenden abdeckt. Dieses oberste Bit im Ergebnis setzen, wenn es im Dividenden auch Eins ist. Dann Subtraktion, d.h. XOR durchführen und Divisor nach rechts schieben. Zum Schluß kann ein Rest im Dividenden-Register stehen bleiben.

Das Beispielprogramm in FORTH ( Listing POLY1 ) reicht nur für 8/16 Bit Wortlänge. Die Division funktioniert in dieser Implementierung nur, wenn Divisor kürzer als der Dividend ist. Deshalb Vorsicht geboten. Mit der Multiplikation kann man die Funktion der Division prüfen, indem man Quotient und Divisor multipliziert und den Rest addiert. Was den Dividenden ergeben sollte.

## Schieberegister

Multiplikation und Division mit Konstanten lassen sich seriell einfach mit Schieberegistern durchführen. Dabei gibt es jeweils die beiden Realisierungsvarianten I und II ( Bild 5 ). Die Schaltungen sind hier zeichne

Bild 4: Division

$$C = \begin{array}{r} a \\ 1100110000 \\ 1 \overline{) 10011} \leftarrow 5 \\ \underline{10101} \\ 10111 \\ \underline{01100} \\ 00000 \\ \underline{11000} \\ 10011 \\ \underline{10110} \\ 10011 \\ \underline{01010} \\ 00000 \\ R = 1010 \end{array} \quad / b = 10011 = 110110$$

generell vereinfacht dargestellt. Sie beginnen mit XOR-Operation mit max Eingängen und Flipflops, die von dem Beginn der Division durch einen Clear Impuls gelöscht werden müssen.

Wird ein Full-Register mit max 4 Wörtern der Länge des Divisors  $q(x)$  als Register benutzt, wo die XORs durchgeführt werden. Das Register ist im ersten Taktperiode nicht in Betrieb. Im Betrieb wird das Divisor mit MSB in das Register und immer MSB in das Register einlesen.

Die Multiplikationsfunktion wird in dieser Anwendung nicht benötigt. Bei den zyklischen Schaltungen ist die Multiplikation nicht erforderlich. Die Multiplikation ist in der Rechnung als Rest in den Schieberegistern sichtbar.

Die Beispielprogramme für die vier Funktionen Listings

```

POLY1 = 17 POLY2 = 177
POLY3 = 17 POLY4 = 177

```

werden in dem mit 16 Bit. Wie man die Funktionen durchführt, die Länge der Eingänge geben, und sind dem für die verschiedenen Polynome und Datenwortlänge nicht verändert. Für praktische Einsatz er komplexer und langsam aber zu Simulation und Erstellung von Testschaltungen, die

## Codes

Zyklische Codes werden nicht durch eine Generationsmatrix, sondern ein Generatorpolynom definiert. Durchweg die Multiplikation erfolgt durch Division des Dividenden durch den Generatorpolynom, wobei der

Bild 3: Multiplikation

$$a(x) * b(x) = c(x)$$

$$a(x) = x^2 + x + 1$$

$$b(x) = x^2 - x^2 + x + 1$$

$b(x) \setminus a(x)$	$x^2$	$x^1$	$x^0$	
$x^2$	1	0	1	1
$x^1$	1	0	1	1
$x^2$	1	0	1	1
$x^1$	0	1	1	1
$x^0$	1	0	1	1

$$c(x) = 1\ 0\ 0\ 0\ 0\ 0\ 1$$

$$g(x) = x^6 + x^5 + x^4 + x^3 + 1$$

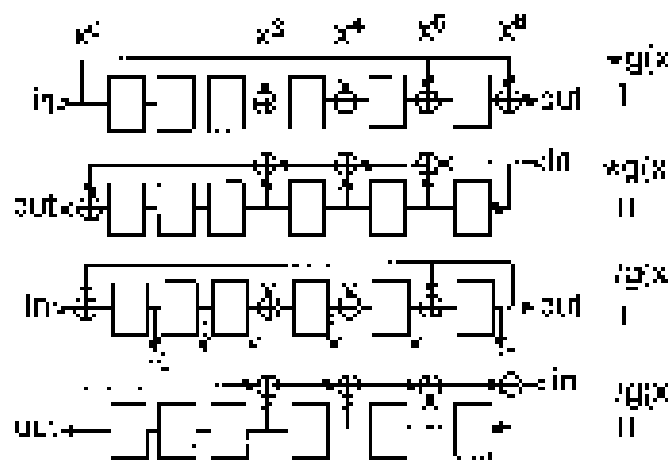


Bild 5: Multiplikation und Division von Konstanten

(Bild 5): Beim Schieberegister ist notwendig, dass so lang wie die Nachricht ist, gefügt werden XOR-Gate, das zur Implementierung als Schieberegister bekannten Bausteins durch einen Systementwerfer bei der ursprünglichen Divisionsanordnung des Coders entspricht. In einem Logiknetz das im nächsten Systementwerfer erscheint.

Realität, als Polynom an der Divisionsanordnung, die in der ersten Zeile dargestellt ist. Die Divisionsanordnung besteht aus gemeinsamen Faktoren der Rest zu berechnen. Der Quotient der dabei herauskommt sind zwei gleiche Bits 5 sein, die direkte Anwendung auf ein 4-Bit'scher Hamming-Code = (3,4). Oben die Divisionsanordnung für den Rest, mit Initialwert 0000. Die Schieberegister mit 4-Bit'scher Invertierung A, dann 5 Taktzyklen, in denen die Nachricht in Schieberegister und Kanal eingelesen wird. Dann wird in Schieberegister B 4 Taktzyklen nötig in denen 4 Nachrichten in Schieberegister übertragen werden, während diese die Transition berechnet. In den letzten 4 Taktzyklen, wenn die Nachricht 0 ist, dann der Rest, die Prüfsumme in den Kanal geschickt. In dieser Schieberegister sind die Nullen gegeben, das ist immer mit dem 0000 von ein Schieberegister kein Prüfsumme mehr.

Die Prüfsumme besteht in Schieberegister 4 sein, dann in 5 Taktzyklen die Prüfsumme angefangen. Das Logiknetz ENCC enthält den Code, der sich wieder durch Vergabe des Polynoms zusammensetzen die Länge aussetzt.

### 3. Variante

Für diesen Code (100101) und von einem Generatorpolynom  $g(x) = x^4 + x^3 + x^2 + x + 1$  in Using PCFA 1 in digitaler Stellung A übertrag, nur der Daten im Schieberegister und in den Kanal. Dann legt man den Schieberegister Stellung B und dann den Rest der Schieberegister sind, das XOR Gate in der Kanal 4-Bit'scher Teile, sich in Stellung ENCC.

### Decoder

Für 4-Bit'scher Code, die Prüfsumme ergibt sich der Hamming-Code. Es besteht an wesentlichen aus 4 Bit'scher, wenn

Die schaltung funktioniert, aber wie die Nachricht nicht sehr elegant. Das Logiknetz 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000.

### Vormultiplikation

Hier ist die Vormultiplikation mit einem 4-Bit'scher Schieberegister, Bild 7 oben. Zuerst die kontinuierliche Schaltung. Hier wird durch 4 Taktzyklen in Stellung A die Nachricht in den Kanal geschickt und danach

$$g(x) = x^2 + x + 1$$

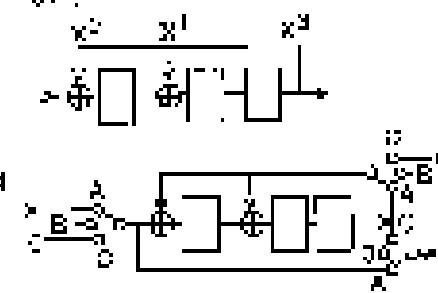
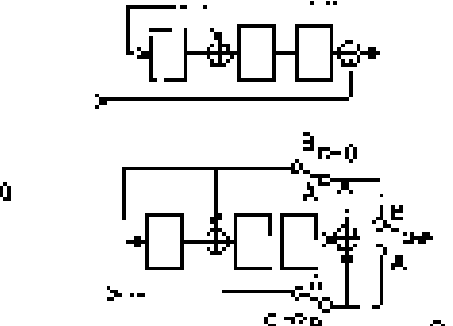


Bild 7:

### Code mit Vormultiplikation



$$h(x) = \frac{x^4 + 1}{q(x)} = \frac{x^4 + 1}{x^3 + x^2 + x + 1}$$

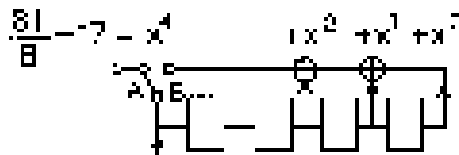


Bild 8: CRC-3 Variante

ung von Mehrbitfehler. Die kompliziertere Codes war jedoch meist ein anderes Verfahren, „Error Trapping“, entwickelt. Man über in einer späteren Ausgabe.

## CRC

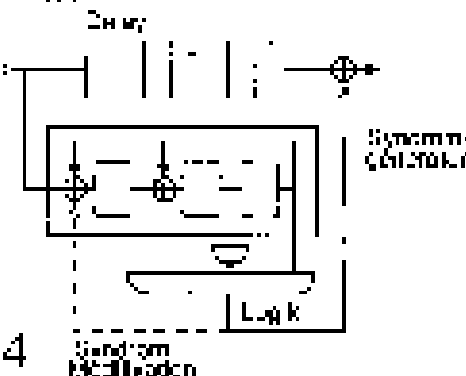
Wenn das alle irgendwie konkret vorkommt bei „Medizin Technisches Glück“ merkt man sich so, nur das man sich dann nicht unbedingt nachrechnen ob der Prüfsumme Null ist und keine Garantie bekommt. Die Generatormultipolynome sind sich dafür auch nicht eignen. Dazu kann man aber auch von CRC die allgemeine Algorithmus für Implementierung in Software übersehen.

## Generatormultipolynome

In Tabelle 2 sind 16 Codes unterteilt in Länge der Nachricht (n), Anzahl der Prüfsumme (k) und Anzahl Fehler (m) die sich mit einem Code beheben lassen. Dabei werden die Codes (1-3) aufgeführt. Das Code der Länge 16 hat die Prüfsumme 4. Wenn man sich die die Länge der Nachricht weißlich kann man.

Bei der (1) ist praktische Verwendung zu kann es sein, dass die Länge der Nachricht (n) aufgeführt werden nicht alle Fehler korrigieren kann.

Bild 9: Meggin-Reorder Diagramm



Da nicht die Wertebereiche weitgehend für ein Byte.

Zwischen man zuerst eine Darstellung und führt sie zu einem Code. Dann man führt den Code mit dem Bytes und ändert die oberste Bit beim Senden. In Empfänger man man ein Zweite Byte sendet, durch 3 Nullen zur Ermittlung einer Länge entgegen kann man die Nachricht durch den Decoder schenkt. Die 3 Variante ist die Verwendung geteilter Codes. Sie soll in einer der nächsten Ausgaben dargestellt werden.

(1) Len. Control „Error Control Coding“ (Prinzip) (1975)

Bild 10: Meggin-Diagramm mit Vollmultiplikation

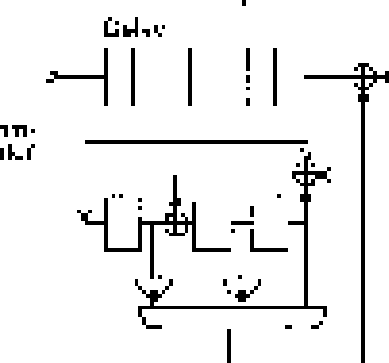


Tabelle 1: Hamming (7,4) Nach [1] & [2]

H7	H4	
00	00	kein Fehler
01	01	
02	02	
04	04	
06	03	
10	06	
11	07	
11	05	1 Bit Fehler

Tabelle 2: Generatormultipolynome

01	1,1	= 0
11	1,1	= 0
15	1,1,2	= 11
15	1,1,3	= 101
15	1,1,4	= 110
11	2,1,1	= 25
11	2,1,3	= 105
02	3,1,1	= 41
02	3,1,2	= 1035

# Scrambler

Einfache LFSRs finden besonders in der Datenübertragung Anwendung. Viele Kanäle können keine langen Folgen von Einsen oder Nullen übertragen, sondern brauchen „Wechselspannung“, damit der Empfänger den Takt resynchronisieren kann. Zusätzlich erwarten Echokompensatoren, die Daten als unkorreliertes Signal.

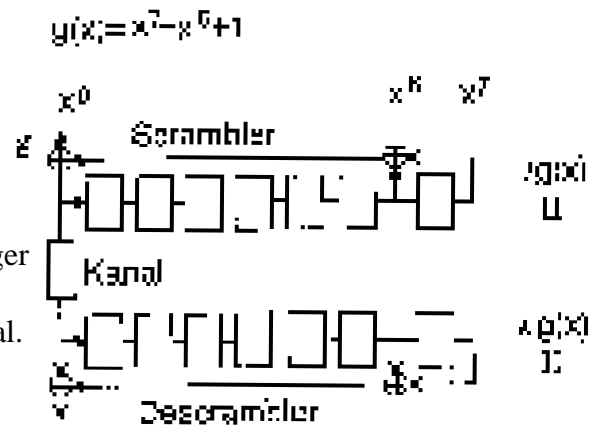


Bild 1: Blockschaltbild

Scrambler im Sender und Descrambler im Empfänger ( Bild 1 ) erfüllen diese Funktion. Der Scrambler ist eine Divisionsschaltung Typ II ( Vgl Artikel „Zyklische Codes“ in diesem Heft ) bei der man die Daten vor dem Schieberegister auskoppelt um Verzögerung in der Übertragung zu verhindern. Wenn die Maschine eingelaufen ist, erzeugt sie keine anderen Daten als die Normalform. Der Descrambler ist eine Multiplikationsschaltung Typ II ( Listing SCRAM )

## Patches

Bild 1 ist nur das Prinzipschaltbild, Anwendungsschaltungen müssen um Hilfslogik erweitert werden.

Erstens muß man den unerlaubten Null-Zustand bei Initialisierung des Scrambler-Registers vermeiden.

Sendet man dann dauernd Datenbits

0, bleibt der Kanal auf Null.

Umgekehrt bleibt der Scrambler im Zustand 1, wenn dauernd Datenbits 1 gesendet werden und alle Bits im Register gesetzt sind.

[1] Bingham „The Theory and Practice of Modem Design“ Wiley 1988

## Auto-Sync

Eine der interessantesten Eigenschaften ist, daß sich der Descrambler nach einiger Zeit selbst synchronisiert. Man muß also nicht mit identischem Registerinhalt die Übertragung beginnen.

## Polynome

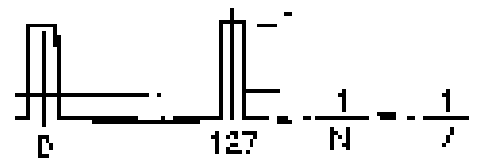
Typische Werte für Anzapfpunkte mit Modem-Anwendung gibt Tabelle 1 [1]. Es handelt sich um m-Sequenzen, speziell solche mit besonders wenigen Anzapfpunkten. Sendet man dauernd Datenbits 0, ist das XOR am Eingang transparent und der Scrambler entspricht dem üblichen PN-Generator.

Wenige Taps sind zwingend erforderlich, denn bei Einzelbitfehlern im Kanal wird der Descrambler die Zahl der fehlerhaften Bits proportional zur Anzahl der Taps erhöhen. Bei 2 Taps werden aus einem Fehler im Kanal schon 3 Fehlern am Ausgang des Descramblers.

Tabelle 1: Polynome für Scrambler

x <sup>3</sup>	x <sup>2</sup>	1		
x <sup>3</sup>	x <sup>2</sup>	1		
x <sup>4</sup>	x <sup>2</sup>	1		
x <sup>5</sup>	x <sup>4</sup>	1		
x <sup>5</sup>	x <sup>1</sup>	1	( 027 1 4 101 )	
x <sup>5</sup>	x <sup>1</sup>	1	?	
x <sup>7</sup>	x <sup>6</sup>	x <sup>4</sup>	x <sup>3</sup>	1
x <sup>6</sup>	x <sup>5</sup>	1		
x <sup>9</sup>	x <sup>8</sup>	1		
x <sup>10</sup>	x <sup>9</sup>	1		
x <sup>11</sup>	x <sup>10</sup>	x <sup>7</sup>	x <sup>6</sup>	1
x <sup>11</sup>	x <sup>10</sup>	x <sup>8</sup>	x <sup>7</sup>	1
x <sup>12</sup>	x <sup>11</sup>	x <sup>9</sup>	x <sup>8</sup>	1
x <sup>13</sup>	x <sup>12</sup>	x <sup>7</sup>	x <sup>4</sup>	1
x <sup>14</sup>	x <sup>13</sup>	1		
x <sup>15</sup>	x <sup>14</sup>	x <sup>13</sup>	x <sup>11</sup>	1
x <sup>16</sup>	x <sup>15</sup>	1	( 022 022011 )	
x <sup>17</sup>	x <sup>16</sup>	1		
x <sup>18</sup>	x <sup>17</sup>	x <sup>16</sup>	x <sup>13</sup>	1
x <sup>19</sup>	x <sup>18</sup>	1		
x <sup>20</sup>	x <sup>19</sup>	1		
x <sup>21</sup>	x <sup>20</sup>	1		
x <sup>22</sup>	x <sup>21</sup>	1	( 023 10111 )	
x <sup>22</sup>	x <sup>21</sup>	1	( 1010 000 )	
x <sup>23</sup>	x <sup>22</sup>	x <sup>21</sup>	x <sup>16</sup>	1
x <sup>24</sup>	x <sup>23</sup>	1		

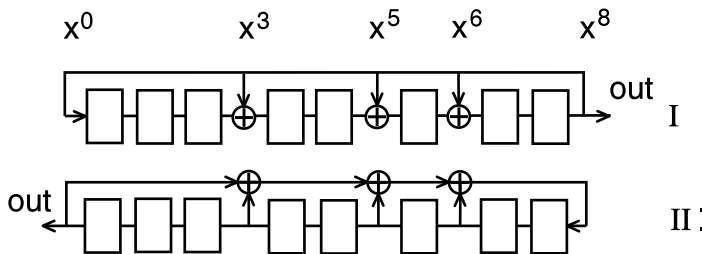
# Zufallszahlengenerator mit Schieberegistern



In Hardware sehr beliebt und auch in Software einfach zu realisieren, da nur XOR und Shifts benötigt werden.

Bild 1:  $g(x) = x^8 + x^6 + x^5 + x^3 + 1$

Implementierung



In Bild 1 ist ein Beispiel in beiden Implementierungsvarianten für das „Linear Feedback Shift Register“ (LFSR) dargestellt. Es handelt sich um Divisionsschaltungen bei denen die Eingangsdaten konstant Null sind und damit das XOR am Eingang entfällt.

Version II ist die Version, die man üblicherweise in der Literatur findet. Man kann sie einfach in Hardware mit MSI-Logik aufbauen. Für sie gelten alle im Text gemachten Eigenschaften.

Version I wäre für Software geeigneter (Vgl. Listings GEN1 = I und GEN2 = II). Zur Äquivalenz vgl. Nachtrag in emb8

generator will man eine möglichst lange Periode. Je höher die Anforderungen in diese Richtung, desto länger das Schieberegister.

## Polynome

Minimal ist ein XOR nötig und diese sehr häufig verwendete Sorte nennt sich „trinomials“. Für 3 XORs; gibt es den Namen „pentanomial“. Generatoren funktionieren nur mit einer ungeraden Anzahl von Anzapfungen.

Tabelle 1: Beispiele m-Sequenzen

## m-Sequenzen

Vor dem Start muß man das Register mit einem Wert ungleich 00h laden. Denn Null ist die magische Zahl („demon state“, „hang-up“), mit der die Maschine zu funktionieren aufhört und die deshalb auch im Betrieb nie auftauchen wird.

Mit geeigneten XOR-Anzapfpunkten („Taps“) wird eine „Maximum Length Sequence“ (m-Sequenz) erzeugt. Mit einem 8 Bit Schieberegister wird der Zyklus die maximalen 255 Zustände haben, da Null ja nicht erlaubt ist. Ist er durchlaufen, fängt der Ablauf wieder von vorne an (Vgl. Printouts von Beispielen in Tabelle 1). Für einen Zufallszahlen-

II-GEN	II-GEN	I-GEN	I-GEN
0 100	0 1000	0 110	0 1100
0 010	0 0100	1 011	0 0110
1 101	0 0010	1 111	1 0011
0 110	1 1001	1 101	1 1101
1 111	0 1100	0 100	0 1010
1 011	0 0110	0 010	1 0101
1 001	1 1011	1 001	0 1110
0 100	1 0101	0 110	1 0101
	0 1010	1 011	1 1101
	1 1101		0 1110
	0 1110		1 1111
	1 1111		1 0111
	1 0111		1 0011
	1 0011		1 0001
	1 0001		0 1000
	0 1000		

Bild 2:

AZF eines 7 Bit PN-Zählens  
Amplitude auf 1,0 normalisiert

Aus jeder m Sequenz MSB oder eine beliebige willkürliche Sequenz direkt ablesen. Die restliche Kombination der Taps ergibt diese:

Tabelle 3:

primale Polynome

0 x<sup>0</sup> + 1

1 x<sup>1</sup> + 1

2 x<sup>2</sup> + 1

3 x<sup>3</sup> + 1

4 x<sup>4</sup> + 1

5 x<sup>5</sup> + 1

6 x<sup>6</sup> + 1

7 x<sup>7</sup> + 1

8 x<sup>8</sup> + 1

9 x<sup>9</sup> + 1

10 x<sup>10</sup> + 1

11 x<sup>11</sup> + 1

12 x<sup>12</sup> + 1

13 x<sup>13</sup> + 1

14 x<sup>14</sup> + 1

15 x<sup>15</sup> + 1

16 x<sup>16</sup> + 1

17 x<sup>17</sup> + 1

18 x<sup>18</sup> + 1

19 x<sup>19</sup> + 1

20 x<sup>20</sup> + 1

21 x<sup>21</sup> + 1

22 x<sup>22</sup> + 1

23 x<sup>23</sup> + 1

24 x<sup>24</sup> + 1

25 x<sup>25</sup> + 1

26 x<sup>26</sup> + 1

27 x<sup>27</sup> + 1

28 x<sup>28</sup> + 1

29 x<sup>29</sup> + 1

30 x<sup>30</sup> + 1

31 x<sup>31</sup> + 1

32 x<sup>32</sup> + 1

33 x<sup>33</sup> + 1

34 x<sup>34</sup> + 1

35 x<sup>35</sup> + 1

36 x<sup>36</sup> + 1

37 x<sup>37</sup> + 1





		x119	x122	1	x170	x174	x200	x204	1	x1221	x1224	x126	x127	1
x26	x26	x22	x21	1	x175	x175	x241	x240	1	x1225	x1225	x130	x209	1
x27	x27	x22	x21	1	x177	x175	x247	x245	1			x1227	x1228	1
		x229	x225	1	x179	x177	x259	x250	1	x1226	x1226	x1201	x299	1
		x229	x227	1			x279	x270	1			x1225	x1229	1
x30	x26	x21	x21	1	x200	x279	x223	x215	1			x1230	x1227	1
		x231	x229	1			x221	x277	1	x1231	x1230	x204	x203	1
x32	x222	x22	x21	1	x202	x279	x247	x244	1			x1232	x1203	1
		x233	x229	1	x203	x282	x222	x217	1	x1233	x1232	x252	x261	1
x34	x227	x22	x21	1			x204	x211	1			x1234	x277	1
		x235	x233	1	x204	x204	x250	x257	1			x1235	x1222	1
		x235	x225	1	x205	x205	x274	x277	1	x1236	x1235	x211	x215	1
x37	x25	x23	x22	1			x287	x24	1			x1237	x1116	1
x38	x25	x25	x21	1	x233	x287	x227	x215	1	x1238	x1237	x1231	x1225	1
		x237	x235	1			x289	x251	1	x1238	x1238	x1234	x1231	1
x40	x223	x221	x219	1	x290	x289	x222	x221	1			x1240	x1111	1
		x241	x228	1	x291	x290	x22	x22	1	x1241	x1240	x1115	x1229	1
x41	x221	x219	x219	1	x292	x297	x280	x228	1			x1242	x1221	1
x43	x222	x222	x227	1			x297	x257	1	x1243	x1242	x1223	x1222	1
x44	x223	x223	x227	1			x294	x272	1	x1244	x1243	x275	x272	1
x45	x222	x222	x222	1			x295	x284	1			x1245	x223	1
x46	x223	x223	x225	1	x296	x294	x245	x247	1	x1246	x1245	x227	x225	1
		x227	x222	1			x297	x251	1	x1247	x1246	x1125	x1200	1
x49	x227	x221	x220	1			x298	x267	1			x1248	x1221	1
		x229	x220	1	x299	x297	x224	x222	1	x1249	x1249	x220	x229	1
x50	x229	x224	x223	1			x2200	x221	1			x1250	x297	1
x51	x220	x226	x225	1	x2201	x2200	x225	x224	1			x1221	x1222	1
		x222	x229	1	x2202	x2202	x226	x225	1	x1222	x1221	x227	x226	1
x53	x222	x228	x227	1			x2202	x224	1			x1222	x1222	1
x54	x223	x228	x227	1	x2204	x2202	x224	x224	1	x1224	x1222	x227	x225	1
		x228	x227	1			x2205	x225	1	x1224	x1224	x1224	x1223	1
x55	x222	x228	x224	1			x2206	x221	1	x1224	x1224	x221	x220	1
		x227	x226	1	x2207	x2205	x221	x227	1	x1227	x1226	x1221	x1220	1
		x228	x226	1			x2206	x227	1	x1228	x1227	x1222	x1220	1
x59	x228	x228	x227	1	x2209	x2209	x2203	x2222	1			x1229	x2223	1
		x220	x220	1	x1110	x1109	x226	x227	1	x1230	x1230	x1222	x1221	1
x57	x220	x226	x225	1			x1111	x1201	1			x1231	x1222	1
x57	x220	x226	x225	1	x1112	x1110	x229	x227	1	x1232	x1231	x225	x224	1
		x223	x222	1			x1113	x1204	1	x1233	x1232	x1204	x2202	1
x64	x221	x221	x220	1	x1114	x1113	x221	x222	1	x1234	x1233	x1231	x2200	1
		x222	x222	1	x1115	x1114	x1221	x1221	1	x1234	x1234	x1231	x2224	1
x66	x222	x227	x226	1	x1116	x1115	x226	x225	1	x1235	x1235	x1228	x2227	1
x67	x222	x228	x227	1	x1117	x1115	x229	x227	1			x1227	x1222	1
		x228	x225	1			x1119	x225	1	x1235	x1235	x1233	x1222	1
x69	x227	x221	x220	1			x1119	x1111	1					1
x70	x225	x225	x224	1	x1220	x1113	x22	x22	1					1
		x221	x225	1			x1221	x1203	1					1
x72	x226	x225	x225	1	x1222	x1221	x223	x222	1					1
		x223	x226	1			x1223	x1221	1					1
x74	x220	x223	x226	1			x1224	x227	1					1

# DTMF

Das „Dual Tone Multi Frequency“ Verfahren war ursprünglich zum Übertragen der Wahlinformation bei Telefonen gedacht. Die Daten werden durch zwei Tonfrequenzen im Sprachband („inband“) übertragen.



Bild 1:  
Die ersten DTMF-Telefone

Ursprünglich in den USA entwickelt [1], ist das Verfahren dort auch unter dem Namen „Touch-Tone“, einem Warenzeichen von AT&T, bekannt. Der Begriff weist darauf hin, daß für den Endanwender dort sein Einsatz mit der Einführung von Tastentelefonen ca. 1964 ( Bild 1 ) synonym ist. Hierzulande ist die offizielle Bezeichnung MFV und außerhalb von Nebenstellenanlagen begann die Einführung erst in den 80er Jahren. Anders als sonst in der Telefontechnik herrscht technische Kompatibilität, da die Bell-Norm von CEPT und CCITT übernommen wurde.

Die heutigen Anwendungen sind vielfältig. Da man mit DTMF auch nach aufgebauter Verbindung im Telefonnetz Informationen übertragen kann, eignet es sich z.B. um mit einem Handsender über akustische Ankopplung von einem Telefon aus Geräte zu steuern. Auch kann man es in anderen Sprachkanälen z.B. Funk verwenden. Da die ICs relativ billig

sind, sind sie ein Ersatz für Modems bei Telemetrie-Anwendung in denen niedrige Datenrate tolerierbar ist.

Im Folgenden seien kurz die Spezifikationen und Realisierungsformen der üblichen ICs für Senden und Empfang angesprochen. Einerseits weil sich an ihnen eine selbstgestrickte Lösung messen muß. Andererseits weil man nützliche Anregungen für eigene Implementierung erhält. Auch wenn man nur einen Sender bauen will, ist es nützlich das Verhalten der üblichen Empfänger zu kennen, da untergeordnete Eigenschaften wie Klirrfaktor sich auf dessen Funktion auswirken können.

## DTMF-Sender

Das Signal besteht aus Sinustönen mit den in Tabelle 1 angegebenen Frequenzen. Es werden immer zwei Töne gesendet, einer aus der oberen und einer aus der unteren Gruppe. Man codiert damit alle 10 Ziffern, die beiden Sonderzeichen \* und #. Ferner die Buchstaben A, B, C, D, auf die der Endanwender an üblichen Telefonen jedoch keinen Zugriff hat.

In Tabelle 2 sind die typischen Anforderungen aufgelistet. Die vorgesehene Preemphase erklärt sich aus den Gegebenheiten im Drahttelefonnetz. Dort werden die höheren Frequenzen durch das Tiefpaßverhalten des Kabels im Ortsnetz

Tabelle 1: Frequenzen

Hz	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

Tabelle 2: Specs Sender

Frequenztoleranz	=< +/-1,8%
Anschwingzeit auf 90%	=< 7msec
Klirrfaktor	=< 5%
Pegelunterschied in einer Gruppe	< 2 dB
Preemphase obere Gruppe	ca. 3 dB



Bild 2: typische Signalform

bedeutet. Man nicht die Details und die Ausdehnung in einem Kanal mit oberem Grenzfrequenz sind nur auf Plausibilität beschränkt. Der Begriff „typisch“ bedeutet allgemein einen Amplitudenverhältnis zwischen zwei Frequenzen. Insbesondere orientiert sich an Beziehung der Amplituden unterschiedlicher Seiten der beiden Frequenzgruppen.

## typische ICs

Die übliche Implementierung ist weiter für zwei 4 Bit DACs durch Suchen die Werte kann man nicht werden. Die Kernliste der DACs ist nicht linear, sondern in das Signal, umgeben in den Klirrfaktor in zwei Tönen. Jeder Zyklus wird aus 20 Samples zusammengesetzt. Der Frequenzfehler ist meist besser 0,7%. Das Verhalten des Reglers wie in Bild 3 und die Spezifikationen sind in Bild 4. Die Verzerrungen liegen meist 30-40 dB unter dem Summenpegel des Signals. In einigen Schichten ist die Belastbarkeit nicht die Summenleistung.

Tabelle 3: Specs Empfänger

gültige Zeichenlänge	>= 40msec
ungültige Zeichenlänge	< 20msec
minimale Pause	100msec
Preemphase	+/-2,5%
Klirrfaktor	-30 ... -3 dBm
Distorsion	-6 ... +8 dB
Twistreject	16 ... 18 dB
DTMF Toleranz	- 6 dB
Dial Tone Toleranz	+18 dB
Noise Toleranz	-12 dB
Teil erweichungsanforderung	U<sup>1</sup>-1
Talk Off	>



Bild 3: typisches Spektrum

simuliert die akustischen Frequenz. In gut verteilten die sehr niedrigen Klankanteile, auch außerhalb des Sprachbands (siehe Bild 4). Es waren aber noch um 20dB im Bereich 4 - 20kHz vorlag. Meist gering im hochgeschalteten 1 oder 2000Hz EC Filter mit die Anforderungen knapp erhaltend.

**Empfänger**

shelle 1 zeigt typische Maßleistungen. Da die Frequenzen eines Telefonsystems im Telefonnetz um ca. +4dB verstoßen werden können, muß die Töne und besonders in getrennt werden. In Praktischer kann das noch zusätzliche sein. Fehlt die normale IC (siehe auch 400) die der Empfängerbereich ist ASIR 2000, 10m.

**Mit Töne**

Im Telefonband bis zu 3000 Hz wird der Wähler (siehe auch 1) der Empfänger (2) auf muß warten, während dieses Signal einigt. Stattdessen steigt von 0dB der Pegel des Wählers hoch, so das DTMF-Signale gleichmäßig ein. In den USA besteht der Wähler aus zwei gleichzeitig gesendeten Tönen bei 1200 und 1600Hz + 25% Erhöhe die Töne 1. Der Rest „Dual Tone Multiple“ in Daten stellen besteht sich meist auf die Unterdrückung genau dieses Frequenz mit speziellen Handapparate in diesen Frequenzen.

In Teilband vom 2000 bis Wähler im Teil bei 4000Hz. In diesem Teilband Hosenlagen ab 1000 Hz bis 4000 Hz Frequenz und nach-

mal als Karaoke eingespeist. In dieser Fall in die „Tand Tone“ Toleranz eine schwarzen Angabe. Sie gibt an, wie hoch ein dritter Einsetzen bei „Beliebig“ Frequenz sein kann, so die Funktion von 0dB. Die meisten Hersteller nur verschrieben bestimmten Seite Frequenz auf 200 - 4000Hz, ungleich damit also eingetragenen Wähler für die Messung der „Maxe“ Töne nach wird typisch werden. Besuchen das mit Filter (2000Hz) mit Verstärkung begrenzt wurde sein. Manche Hersteller gehen nicht nach „DTMF“ Parameter von 1000 Maximum Intervalle mit 500Hz und endet mit Oberwellen kommt, ist diese Angabe allerdings wenig brauchbar.

**Talk Off**

Der Empfänger kann zwei Fehler machen, er kann eine plötzliches Signal ansprechen. Oder er kann in Stunden im selben Bereich zu hören, wenn nach ihm zu sein kann.

Die eine Charakteristischer Hinweis von 1004 auf 1 Fehler auf 1000 gesendete Zeichen auftreten. Für einen Ton, normal mit 1000 Zeichen Laufzeiten von Stunden um auf eine will. Verfügen Grenzen zu kommen.

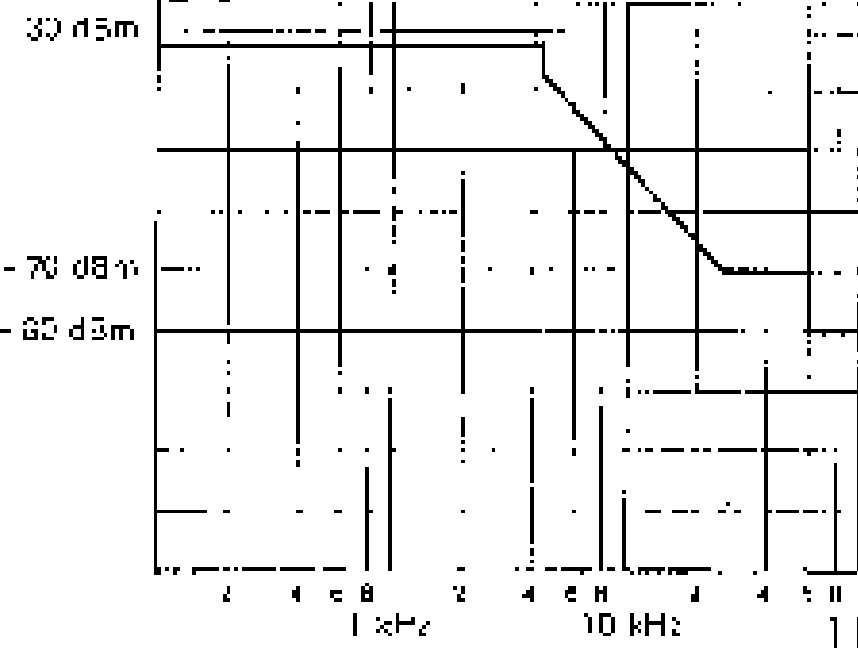
Insbesondere bezüglich Sprachseite liegt man allgemein vor. Besondere und Fallbedingungen als 1000 Zeichen Wähler

sehr viele der offensichtlich kann Sprache einseitig DTMF-Signale (Töne) während der DTMF. Jedoch aus zwei Standard besteht, in Sprache mehrheitlich. Einmal für Sprachschleife und der Länge, der die Übergabezeit im geschalteten Sprachband, insbesondere im Bereich 100 - 2000Hz. Nachweise, gen. Um ihm jedoch dazu, daß die Klankanteile sehr so heißt DTMF-Sender hat immer Empfänger zu Nachbarn den Ohren kann.

Als allgemein verfügbarer Testman ist für Sprachschleife zum mehr bei 2000 Hz. Verband UIC/ETI zeigt, die 30 Minuten Gegenstromschleifen mit. In der Welt messen zum Band wird einige als 30 Fehler als akzeptiert angesehen. Meist schon gut normalerweise nach der die DTMF-Sprache-IC in Tests dem, sprechen. Maßzahl ist eine Hand-Rechnung von Mittel für ein weniger als 1 Fehler typisch sein sollen. Bei Schwerefall 1 S&T) besteht von 2 - 3 Fehler im 1000 Perioden einem seiner K&A. Arbeiter. Mit dem Band in 1000 Hz, die Gesamtsumme der Fehler unter Gleichzeitenschwankungen, 1000 häufige Frequenzfehler im Signal) am mehrwertigen Amplitudengang. Die Größe war mit 300 DM auch nicht eben gering.

J) Schecker „Post Office calling with frequency error“ in „Electrical Journal“ 37 (1960)

Bild 4: Grenzen für Oberwellen wie von IECU erhalten gelandet



# DTMF-Sender in Software

Mit einem D/A-Wandler und ca. 400 Byte Programmspeicher kann ein 8 Bit Controller senden. Er ist allerdings mit der Signalzeugung zu 100% ausgelastet.

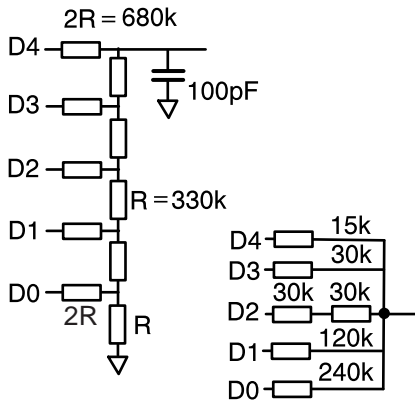


Bild 1 & 2: Wandlernetzwerke

Von absoluter Wichtigkeit ist, daß die Frequenzen auf +/-1,8% eingehalten werden. Auf Preemphasis kann man in Anwendungen außerhalb des öffentlichen Telefonnetzes meist unbesorgt verzichten. Auch die genaue Dauer des Tastendrucks ist in den üblichen Wähleranwendungen unkritisch und sollte nur größer 80 msec sein. Wie sehr man Klirrfaktor optimiert ist Geschmackssache, soweit man nicht vorbestimmte Vorschriften zu erfüllen. Viel Energie von 1633 - 3400 Hz und kurze Signaldauer plus eventuell schiefe Frequenzen können den Empfänger allerdings dazu ermuntern Sprache zu vermuten und nicht anzusprechen.

## DAC

Im günstigsten Fall steht ein im Controller integrierter 8 Bit R2R DAC zur Verfügung. Wenn nicht, realisiert man ihn extern an einem Port für 6 - 8 Bit Auflösung als R2R-Netzwerk ( Bild 1 ). Um Platz zu sparen bieten sich handelsübliche Widerstandsnetzwerke an. Dabei hat man allerdings nicht Zugriff auf das optimale Widerstandsverhältnis 1:2. Für 5 Bit ist auch die Variante in Bild 2 sinnvoll.

$$t_{cyc} = \frac{1}{3,579MHz * 0,5}$$

$$f_{sin} = \frac{1}{T * N * t_{cyc}}$$

## Programm

Die Sinuswerte werden Tabellen entnommen. Da die Wandler lineare Kennlinie haben, kann man die beiden Signale in Software addieren, bevor man sie auf dem DAC ausgibt. Die Amplitude wird deshalb für einen 8 Bit D/A-Wandler z.B. nur 127d betragen. Wenn man Anhebung vorsehen will, haben die Sinuswerte der unteren und oberen Frequenzgruppe unterschiedliche Aussteuerung. Z.B. mit den Maximalwerten 108d und 147d für 2,6dB Preemphasis.

Normalerweise kann man Verzögerungen durch Zeitschleifen in Software bis auf die Zykluszeit des CPU t<sub>cyc</sub> genau realisieren. Hier wird ein 3,6MHz Quarz verwendet, der durch 2 geteilt die Taktfrequenz eines 68HC05 Controllers ergibt. Die Frequenz des erzeugten Sinus entspricht der Formel in Bild 3. Dabei ist N die Zahl der CPU-Zyklen aus denen sich die Samplezeit zusammensetzt. Sie muß für alle Sinussignale einheitlich sein. T ist der Umfang der Sinustabelle ist. Ihre Länge bestimmt die Frequenzteilung und ist damit für jeden Sinus verschieden.

Das Programm ( Listing DTMF.F05 ) tastet somit zyklisch die beiden Sinustabellen ab, addiert die beiden ausgelesenen Werte und gibt sie auf dem D/A-Wandler aus. Nach 80msec wird das Unterprogramm beendet. Da ein 8 Bit Zähler für diese Verzögerung nicht ausreicht, wird die untere Sinus-Frequenz als Vorteiler verwendet.

## Tabellen

Zur Bestimmung geeigneter Werte schreibt man sich am besten ein einfaches Programm in BASIC oder einem FORTH mit Float. Hier wurde F-PC für das Listing DTMF.SEQ im Anhang verwendet.

Bild 3: Formeln für 68HC05

Man gibt N von 1 und berechnet solange die 8 Bit DAC Sinuswerte wie die Länge des Sinusbeleg T. Oben schließlich wird T ein Teiler sein. Man wählt T aber geeignet auf- oder abwärts und kann mit diesem Wert die erst benötigte Frequenz zu der nächsten Datenreihe über die Datenbreite an möglichen Jährgängen über hat. Ist dies für einen Wert von N zu klein 8 Sinusfrequenzen der Fall, kann man eine handlichere Kopfgrenze von 3 oder 4 gefunden.

Das Programm führt zu Beispiel DTMF die Suche durch einen Sinusbeleg möglichen N durch und druckt die handlichere Werte aus. Der Befehl PRINT zeigt für ein N die beiden Sinusabweichungen. Bei Bedarf kann man die Tabelle und legt sie in einem Datei in die N=50-100 ergibt LIST 50,51,52,53,54,55,56,57,58. T innerhalb von N=58 findet sich wegen der geringen Auflösung keine Werte mit abnehmender Frequenzabwärtigen mehr. Der Laufzeitbedarf der Software setzt bei einer CPU eine Grenze bei ca. 60. Somit hat man hier nur eine mögliche Samplefrequenz von Aussehen Speicherbedarf der Tabelle für 200 Byte für N=68. Der Rest der Software belegt weitere 135 Byte.

## Abmagerung

Vereinfachungen sind möglich, wenn man beispielsweise nur viele Anwendungen mit alle DTMF-Ziffern benötigt. Wenn man die Signale 8, 0, 1, 2 nicht verwendet, entfällt die 1639 Hz Port. Mit einem 8 Bit DAC-Wandler sind die Tabellenwerte nur noch 4 Bit hoch und man kann sie in Bytes speichern. Als die Daten für die obere Frequenz von Bild 3 sind die Werte in Tabelle 3-3 vgl. [1]. Man wird in diesem Fall nur noch einen Signal-Raumabstand von ungefähr 25dB mit einem Zähler von Bild 1 haben. Mit 6 Bit beginnt man dann Tabellen bei ca. 100k.

[1] Robert Horner, "Telephone Dialing Techniques using the MC6805", Motorola 4/8/81

# DTMF-Messung

Was man baut muß man auch prüfen. Das Gemisch von zwei Sinustönen ist aber mit üblichen Geräten nicht leicht zu messen. Abhilfe schafft eine einfache Hilfsschaltung.

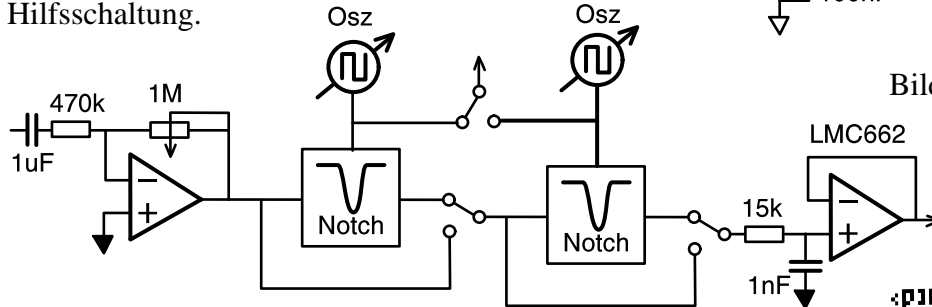


Bild 1: Blockschaltbild

Zwei durchstimmbare Bandsperrfiltern („Notch“) erlauben alle wesentlichen Messungen ( Bild 1 ). Man kann damit jeweils einen der beiden Sinustöne eliminieren und vom anderen Amplitude und Frequenz messen. Oder beide unterdrücken und sich das Restsignal ansehen.

## Schaltung

Verwendet werden dazu zwei SC-Filter LTC1164. Damit kann man bequem steiflankige Filter bauen, die über Taktfrequenz abstimmbare sind. Möglich wären 4 kaskadierte Notch pro IC. Verwendet wird nur einer um die Dämpfung im Durchlaßbereich niedrig zu halten ( Bild 2 ). Im Fädeldrahtaufbau liegt der Noisefloor bei -45dB, was genügt. Wesentlich ist der Tiefpaß am Ausgang der Filter, der die SC-Schaltspikes eliminiert. Dabei

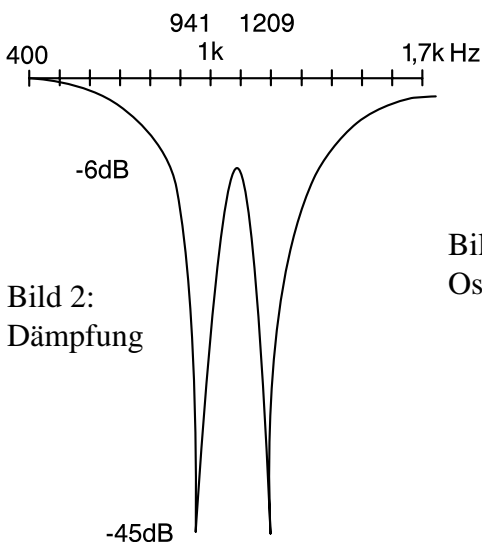


Bild 2: Dämpfung

solte kein unbenutzter OP der SC-Filter verwendet werden, weil diese alle von Schaltspikes verseucht sind. Nützlich ist auch das LC-Filter in der Stromversorgung, das den Taktgenerator abtrennt. Sowie die Zuführung des Taktes an die analogen ICs durch ein geschirmtes Kabel. Ohne diese Maßnahmen liegt man bei nur -30dB. Eine geätzte Platine mit Masseflächen würde weitere Verbesserungen bringen. Die Filter haben die in Bild 3 gezeigte UAF-Struktur. Die Güte könnte durch Ändern des 47k Widerstandes noch erhöht werden.

Mit einem der unbenutzten OPs der Filter-ICs kann man den einstellbaren Verstärker am Eingang aufbauen. Auch bei 10V Versorgung

Tabelle 1:  
Frequenztoleranz 1,8%

684	697	709
750	770	783
836	852	867
924	941	958
1187	1209	1230
1311	1336	1360
1450	1477	1503
1603	1633	1662

Bild 4: Oszillator

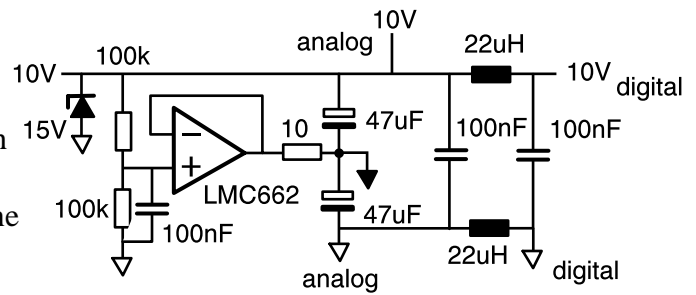
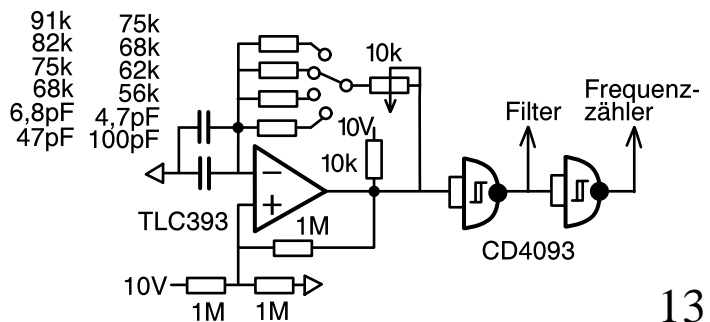


Bild 5: Stromversorgung

speichern, muß das Signal auf etwa 2V<sub>eff</sub> begrenzt werden, weil interne Signale des FAD-Converters sonst gehen können. Auch wichtig ist natürlich für gute Aussteuerung Signal, wenn das Signal nicht im Bereich von 0 bis 1V liegt.

Die Mikrocomputer des Nachbau kann man durch einen anderen ersetzen. Bei bestimmten Versionen der Arduino Boards läßt sich der verwendete ATmega-Mikrocontroller mit einem anderen austauschen. Diese sind durch einen I2C-Bus angeschlossen (Bild 4). Die Schaltung wird zweimal benötigt. Die genannten Bauteile sind deshalb doppelt angegeben. Damit sind die Frequenzwerte in Anlehnung an eine Frequenzliste gewählt. Empfehlenswert ist außerdem die Logik-ICs.

Es werden jeweils die 4 Teilfrequenzen mit einem Oszillator und sind dann mit einem einstellbaren Verstärker über 1,5V<sub>eff</sub> begrenzt einstellbar. Mit handelsüblichen Widerstandswerten für die Taktfrequenz von nicht mehr als 100kHz. Gedacht ist mit Hilfe der Frequenzwerte von Tabelle 1 eine 1000 Hz.

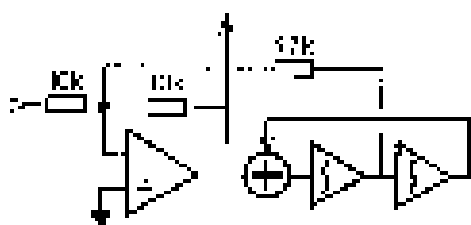


Bild 3: Notch Filter

Allerdings kann man die Filter verstellen und den weiteren Frequenzbereich über die 4 Frequenzen durch die Wandel-Teilnehmer weniger Teilnehmer um nur 100Hz auszuweichen. Im Leistungsanfallfall, was man nicht zu klein mit dem Filter der schwingenden Frequenz.

Die Spannungswandlung an Bereich von 7 - 10 V sind stellen sich. Die 9V Batterie ist wegen die Verwendung von Kommutierarten eine geringe Leistung. Die Stromaufnahme liegt mit hoch ausstrahlend niedrig. Bild 3 zeigt die Erzeugung der Kommutierarten mit 10k. Die 10k sind Widerstand, welche die Schwingung durch kopierbare Last.

### Messung

Für weiter messung 4 MHz werden gespült werden die so gemittelt sein sollten, die 4k 3

Die Frequenzen sind aufpassen. Hier werden die Frequenzen 0 - 6, 8 verwendet. Dabei wurde der 32k 10k mit 10k 10k als Filter benutzt. Die Frequenzen liegen, was bewirkt, die in der 10k 10k Interessanz ist. Vorzeichen mit Ausrichtung von beiden DTMF-Frequenzen 1 10k 10k 2 7 10k 10k 10k 10k.

Tabelle 2: Noise-floor

8 Bit	-47 dB
7 Bit	-50 dB
6 Bit	-25 dB
4 Bit	-30 dB

## Crossassembler PIC 16C5x

PAGE-0 PAGE-1 PAGE-2  
PAGE-3

Warum sollte man für eine CPU einen Postfix-Assembler schreiben? Verbesserte Produktivität bzw. Bequemlichkeit ist die Antwort. Ein eigener Assembler erlaubt es die Funktionen genau so zu implementieren wie man sie gewohnt ist. Damit kann man mit ihm schneller arbeiten und macht weniger Fehler. Die Opcodes sehen so aus wie man sie gerne hat. Hier z.B. sieht der PIC wie ein 6502 aus. Das erlaubt es einem Source die man für andere CPUs geschrieben hat, ohne große Änderungen zu übernehmen.

Die üblichen Initialisierungen sind in dem Befehl INIT zusammengefaßt.

### Daten

Analog zum Zeiger T>C existiert der 16 Bit Pointer

T>ZV

der auf die nächste frei Variablen-Adresse zeigt. Er wird von

n ZVARIABLE <name>

benutzt um Variablen zu definieren. Der Befehl erwartet die Übergabe der Zahl n die die Anzahl der Bytes die zugeteilt werden sollen angibt. Register und I/O haben eigene Namen vordefiniert:

```
RTCC
PRG-CNTR
STATUS
FSR
PA \ Port A
PB \ Port B
PC \ Port C
```

Auch die Flags im ProzessorStatus wurden als Bitkonstanten definiert:

CARRY ZERO

Man kann natürlich auch beliebige neue Opcodes als Makros definieren. Oder dem Assembler leicht neue Funktionen beizubringen. Die werden oft nur für ein Projekt benötigt, sind dort aber sehr nützlich. Ferner lernt man beim Schreiben des Assemblers und Disassemblers den Befehlssatz sehr genau kennen. Das hilft einem, ihn unmittelbar danach mit voller Wirksamkeit einzusetzen.

O>C

vorgibt und der in diesem Fall 4k Byte lang sein sollte. Da der PIC 12 Bit Opcodes hat, werden diese in big-endian als 16 Bit gespeichert. Als Zeiger auf einen freien Speicherplatz dient der 16 Bit Pointer

T>C

er wird von den Opcodes durch dem nochmal definierten Befehl , benutzt, mit dem man nun Worte im Programmspeicher ablegen kann. Wenn das Programm fertig compiliert ist, kann man den Zeiger durch

RES>C!

auf die Position des Resetvektors weiterbewegen und dort den Sprungbefehl compilieren. Der Adreßbereich ist bei einigen CPUs durch Banking erweitert. Dafür ist die Einstellung der Page vorzunehmen:

### Initialisierung

Auswahl der CPUs erfolgt durch die Befehle:

16x54 16x55 16x56 16x57

Dadurch weiß der Compiler z.B. welcher Speicherbereich zulässig ist. Für den Programmspeicher des Targets dient ein Buffer im RAM, dessen absolute Startadresse die Konstante







# V24-Schnittstelle

## Teil 2

Über rote und unrotliche Schaltungen, wenn für Komp-Systeme.

Anleitung zum beschriebenen Aufbau einer RS232C-Verbindung und verbindet zum die Steuerung der Handhabung. Bei geeigneter Verbindung (siehe 1) ermöglicht die ETS2022 bzw. VTR2022 (Analog) -Analog automatisch. Die unrotliche (rotliche) wird vom Terminal als empfangen, was sinnvoll ist. Hier wird auch die RXD-Leitung inaktiv gehalten. Es ist denkbar, daß das Terminal LCD aktiv sein will. Dann muß man es z.B. über Widerstand auf 5V legen. Der Fall ist aber an sich nicht beachtlich, wieder mal.

### Stromlaufschaltungen

Bild 3 zeigt die richtige Lösung mit dem MAX232. Die Schaltung ist nicht, daß man einen Elkos-10V und 10V angeben kann, was für manche Analogschaltungen in Frage ist. Der unrotliche Teil, wiederum kann man einen +10V einen Leistungs MOS-FET ausprägen, um die Schaltung.

Bild 4 stellt die Stromversorgung dar, die wichtig ist. Bitte, Stromlaufschaltung. Sowen, man keine langen Kabel führen will, genügt es. Die schaltete Masse ist, daß die Schaltung.

Bild 3:

Speisungsschaltung

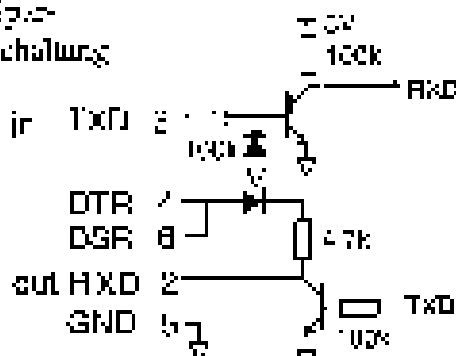
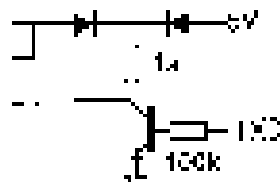


Bild 4:

Mit This spannung



leistung von den Handhabungsträger abgetrennt wird. Wenn die Terminalabgrenzung diese nicht richtig identifiziert, funktioniert der Sender nicht. Man kann über Diode eigene Spannung zuführen, aber wenn man nur 5V an Verfügung hat, wird der Arbeitseinsatz ungenügend.

### Trickschaltungen

Bei einigen Anwendungen muß über die V24 auch Stromversorgung möglich sein. Z.B. bei Geräten, die in Normalbetrieb nur geringe Funktionsleistung benötigen, aber für Einzelarbeiten über einen Laptop anschließen will. Man legt dann nicht funktionierende Geräte in Betrieb, positive Energie kann, dann kann man ein Kabel auf einen 10V Versorgungsgenerator. Bild 5. Mit einem Spannungsregler kann man sich in der Terminal bis zu 10V bei 5V umwandeln.

Für weitere eigene Trick-schaltungen kann man sich an der Spezifikation stellen, 1. Installation.

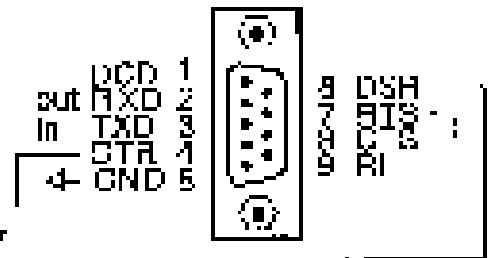


Bild 1: Durchschaltung

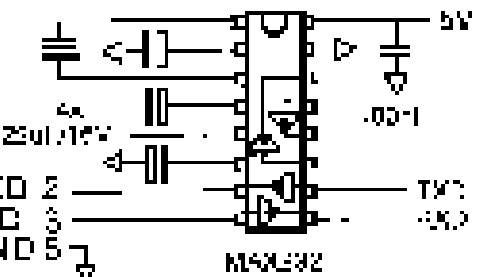


Bild 2: Schaltung mit MAX232

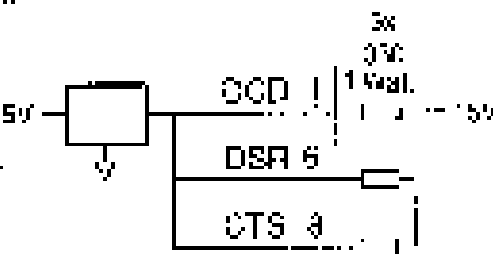


Bild 5: Stromversorgung aus Schnittstelle

Tabelle 1: Spez

PARAMETER	SPECIFICATION	COMMENT
Driver Output Voltage		
0 level	+5V to +15V	With 3-7kΩ load
1 level	-5V to -15V	With 3-7kΩ load
Max. output	+25V Max.	No Load
Receiver Input Thresholds (data and clock signals)		
0 level	+2V to +25V	
1 level	-2V to -25V	
Receiver Thresholds (RS, DSR, DTR)		
On level	+3V to +25V	Driver Power
Off level	Open Circuit or -3V to -25V	Off Stand-By or Drive
Receiver Input Resistance	3kΩ to 7kΩ	
Driver Output Resistance		
Power at condition	300mW Min.	V <sub>DD</sub> = +5V
Driver Rise Time	80ns Max.	3kΩ = R, C <sub>L</sub> = 200pF
Signal rate	Up to 20kbit/sec	
Cable Length	50m Max. Recommended Max. Length	Longer cables permissible if C <sub>L</sub> ≤ 200pF