MEA 8000

Eines der wenigen europäischen Sprachsynthese-ICs. Wenn man digitale Formantsynthese z.B. auf einem Controller implementieren will auch heute noch ein guter Ausgangspunkt.

Flanagan 1972 [1]: „Digitally simulated formant synthesizers - implemented either by programmed operations in general-purpose computers or as special purpose digital hardware - have been used in a variety of forms. Analog hardware synthesizers, controlled by digital computers, have over the past had even more extensive use. Digital implementations, however, have distinct advantages in stability and accuracy, and current advances in digital circuitry make commitment to full digital operation irresistable“. Der Trend ging in den 70er Jahren dann jedoch wegen der günstigeren digitalen Realisierung weg von traditionellen Formant-Synthesizern hin zu LPC.

IPO

Andererseits war bekannt daß LPC für bestimmte Laute nicht gut geeignet ist. Problematisch für manche europäische Sprachen wie z.B. französisch. Das niederländische „Institute for Perception Research“ (IPO) setzte weiterhin auf Formant und baute 1977 einen Synthesizer genannt „Voice Response Unit“ VRU aus MSI-ICs. Ihm folgte 1979 eine verbesserte Version die als 16 Bit 2901 Bit-Slice Rechner ausgeführt wurde [2]. Mit etwa 9 kg Gewicht und 50 Watt Stromverbrauch noch nicht marktfähig. Aber es wurden für Tests 5 Exemplare produziert. Gleichzeitig entstand Software um Sprachdaten auf Minicomputer geeignet zu codieren.

Philips

Der Entwicklungsstand war also recht fortgeschritten als sich Philips ELCOMA („Electronic Components and Materials“) entschloß daraus ein IC zu machen das 1982 verfügbar wurde. Technisch mustergültig gelöst. Trotz 5V NMOS nur 30mA Stromverbrauch. Da man keinen echten D/A-Wandler integrieren konnte war man wie die US ICs auf PWM beschränkt. Sorgte hier aber per upsampling dafür daß extern ein simpler LC-Tiefpaß genügte.

Einen 16 Bit Rechner auf einen Chip zu schrumpfen resultierte aber in 30mm^2 Chipfläche und einem Listenpreis 1984 von 34DM / 100 Stück. Für Consumeranwendungen nicht recht attraktiv.

Ein weiteres Problem war daß die Erzeugung von Sprachdaten erstmal nur durch Philips bzw. Valvo in Hamburg auf einer VAX11/750 erfolgen konnte. Zwar wurde bald darauf angekündigt daß man an einem portableren System für Endanwender arbeite [3]. Was aber 1985 für das Nachfolge-IC PCF8200 als OM8210 rauskam war eine Leiterplatte mit Software für den HP9816S Personal Computer den der Kunde bei HP zusätzlich hätte kaufen müssen. Die Analysesoftware forderte letztlich einen Minicomputer, ein IBM-XT mit 8087 wäre kaum komfortabel gewesen.

Der kommerzielle Erfolg des ICs scheint sich in Grenzen gehalten haben. Die französische Firma TMPI („Techni-musique & parole informatique“) hat Zusatzkarten für einige 8 Bit Homecomputer (Thomson, Amstrad, Oric) angeboten und scheint in Frankreich recht beliebt gewesen zu sein.

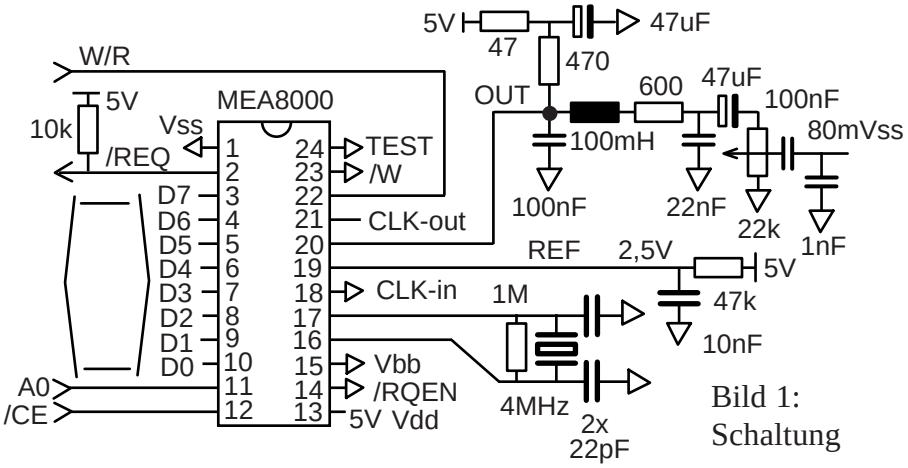


Bild 1: Schaltung

Bild 3: Frequenzgang des Filters [5]

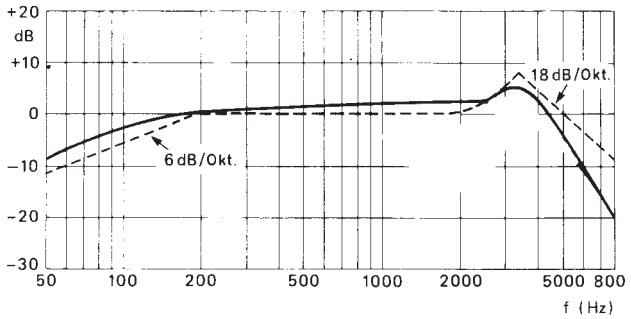
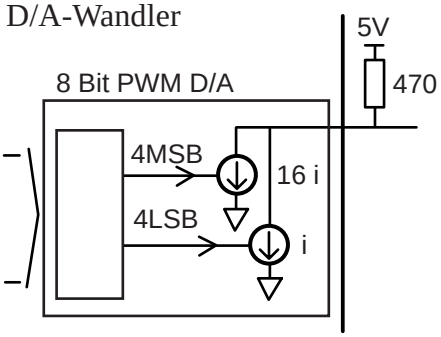


Bild 2: D/A-Wandler



[1] Flanagan „Speech Analysis Synthesis and Perception“ Springer 1972

[2] van Essen and Willems, 1978

[3] Sickert „Hohe Sprachqualität - geringer Speicherbedarf“ Elektronik 4/1984

[4] „MEA 8000 voice synthesizer: principles and interfacing“ Application Note Philips Technical Publication l0l

[5] Bierlaagh „Sprachsynthesizer MEA 8000 Applikationshinweise“ Valvo 1983 (vermutlich Übersetzung von [4])

[6] Sickert „Automatische Spracheingabe und Sprachausgabe“ Markt & Technik 1983

Schaltung

Die digitale Seite ist buskompatibel (Bild 1). Der /REQ-Pin ist ein /IRQ der neue Daten anfordert.

Das Ausgangssignal wird durch lineare Interpolation auf 64kHz hochgesetzt (Bild 4) und dann über einen 8 Bit PWM ausgegeben. Dieser ist als Parallelschaltung von zwei 4 Bit PWMs die Stromsenken ansteuern ausgeführt (Bild 2). Damit ist die Frequenz so hoch daß man ein LC-Filter verwenden kann. Der Serienwiderstand zum Stabkern wird so ausgelegt daß eine leichte Höhenanhebung erfolgt (Bild 3).

Zugriff

Das Statusregister (Tabelle 2) enthält in Bit 7 nur das REQ-Bit das

Tabelle 1: Register

```
A0 W/R
0   1  write data register
1   1  write control reg.
1   0  read status register
```

Tabelle 2: Status Register

```
Rxxxxxxx
^ REQ
```

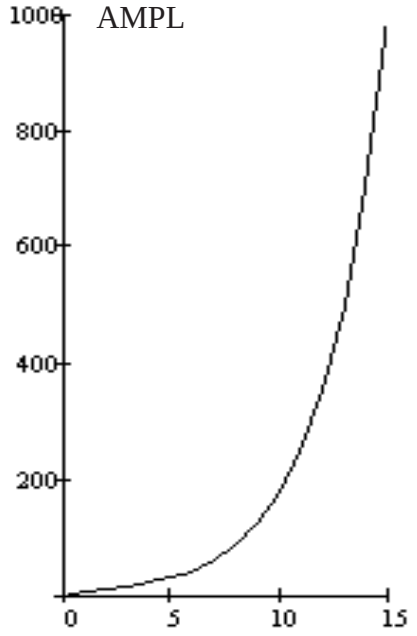
Tabelle 3: Control Register

```
xxxUVVWW
   ^   STOP: 1 STOP
             0 don't care
    ^^ CONT:  00 don't care
              01 don't care
              10 SLOW-STOP
              11 CONTINOUS
  ^^/REQ-Pin
              00 don't care
              01 don't care
              10 inaktiv
              11 aktiv
```

Tabelle 4: Codierung

	PITCH Hz	BWx Hz	FM3 Hz	FM2 Hz	FM1 Hz	AMPL G	PI Hz/8msec	FD msec
00	0	726	1179	440	150	0,000	0	8
01	2	309	1337	466	162	0,008	1	16
02	4	125	1528	494	174	0,011	2	32
03	6	50	1761	523	188	0,016	3	64
04	8		2047	554	202	0,022	4	
05	10		2400	587	217	0,031	5	
06	12		2842	622	233	0,044	6	
07	14		3400	659	250	0,068	7	
08	16			698	267	0,088	8	
09	18			740	286	0,125	9	
0A	20			784	305	0,177	10	
0B	22			830	325	0,250	11	
0C	24			880	346	0,354	12	
0D	26			932	368	0,500	13	
0E	28			988	391	0,707	14	
0F	30			1047	415	1,000	15	
10	32			1110	440		**unvoiced**	
11	34			1179	466		-15	
12	36			1254	494		-14	
13	38			1337	523		-13	
14	40			1428	554		-12	
15	42			1528	587		-11	
16	44			1639	622		-10	
17	46			1761	659		-9	
18	48			1897	698		-8	
19	50			2047	740		-7	
1A	52			2214	784		-6	
1B	54			2400	830		-5	
1C	56			2609	880		-4	
1D	58			2842	932		-3	
1E	60			3105	988		-2	
1F	62			3400	1047		-1	
...								
FF	510							

Bild 6: Nichtlineare Kennlinie



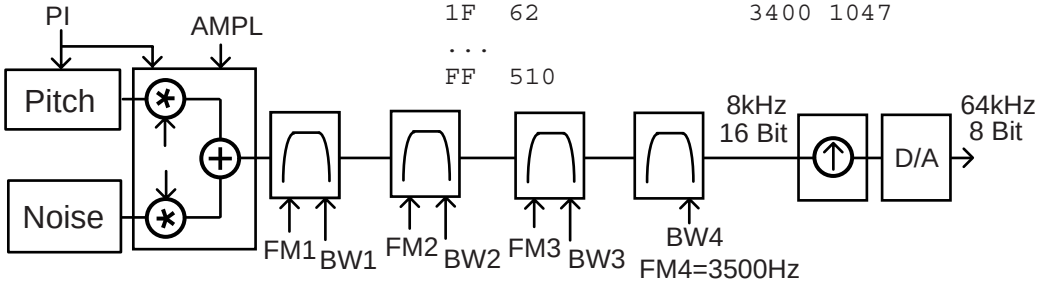


Bild 4: Vokaltrakt

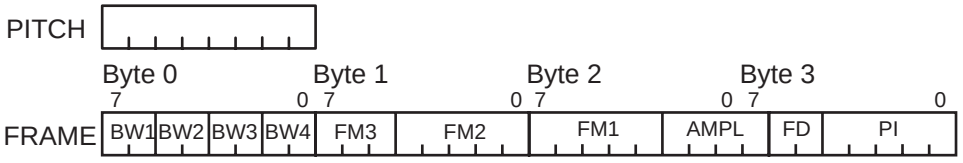


Bild 5: FRAME & Startwert PITCH

dem invertierten Pin in Bild 5 entspricht. Der Pin kann im Status-Register (Tabelle 3) gesperrt werden.

Ansonsten dient dieses Register hauptsächlich zur Übergabe von STOP, SLOW-STOP und CONTINOUS Befehlen durch schreiben entsprechender Bitmuster.

FRAME

Das 32 Bit Datenwort enthält entsprechend Tabelle 4 codiert die Einstellungen für den Vokaltrakt (Bild 3).

Die Filter werden über Mittenfrequenz FMx und Bandbreite BWx eingestellt.

Der Wert in PI enthält als 2er Komplement codiert den Wert der alle 8 msec zum Pitchregister addiert wird. Diese 31 Codeworte legen also „voiced“ fest, ein 32. Codewort schaltet auf „unvoiced“ um.

Die Kennlinie von AMPL für die Signalquellen ist nichtlinear (Bild 5). Der Wert bezieht sich auf die aktive Signalquelle, der andere Kanal ist deaktviert. Die Auswahl geschieht wie gesehen über PI .

Der Einstellung FD „Frame Duration“ schließlich legt die Dauer des Frames fest.

Datenpumpe

Für normalen Betrieb (Bild 7) genügt der STOP Befehl. Ihm folgt am Anfang der Sprachausgabe die Übergabe des Startbytes PITCH ins Datenregister das dann in ein Pitchregister durchgereicht wird. Codierung entspricht Tabelle 4.

Danach werden werden kontinuierlich 32 Bit FRAME Datenworte übergeben. Dabei wird im letzten gültigen Datensatz bereits AMPL=0 gesetzt so daß Reduzierung des Sprachpegels eingeleitet wird.

Danach folgt ein allerletzter Rahmen mit AMPL=0 und erst dann das harte Abschalten mit STOP.

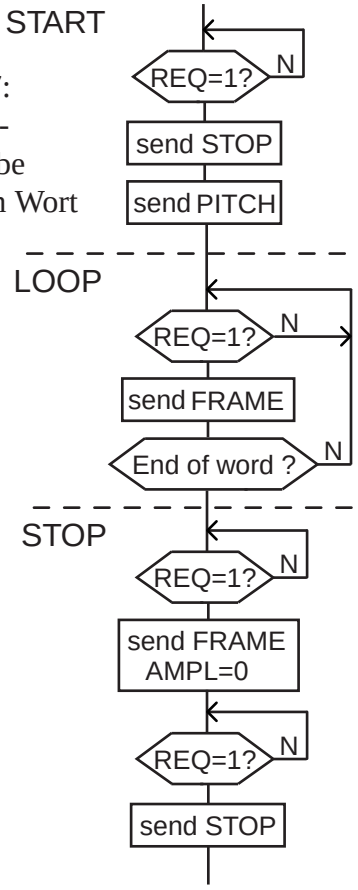


Bild 7: Datenausgabe für ein Wort

Simulator

Eine Nachbildung in Hochsprache ist eine nötige Vorstufe für echtzeitfähige Implementierung in Assembler.

Diese Version erfordert einen 16 Bit Controller mit schnellem 16/32 Bit Multiplizirer. Der Simulator entsprechend ein 16 Bit FORTH.

Nicht alle Details des MEA8000 wurden veröffentlicht. Zweifelsfälle muß man durch Vermutungen ausfüllen. Kann man später durch Vergleich des Verhaltens von IC und Simulator klären.

Filter

Das Flußdiagramm der VRU (Bild 8) hatte konventionelle Allpol-IIR Filter [7] und unterschiedlich angeordnete Sättigungslogik („N“). Diese wurden für das IC in eine Variante geändert bei der man Mittenfrequenz und Bandbreite getrennt ändern kann (Bild 9). Anhand Tabelle 4 ergeben sich 48 Werte für die Frequenz und 4 für die Bandbreite (Tabelle 5, 6) die man in zwei Tabellen zwischenspeichert. Man kann direkte die Skalierung aus [7] verwenden (Bild 10). Negative Vorzeichen für F in Tabelle 5 werden Bit 15 = 1 gekennzeichnet.

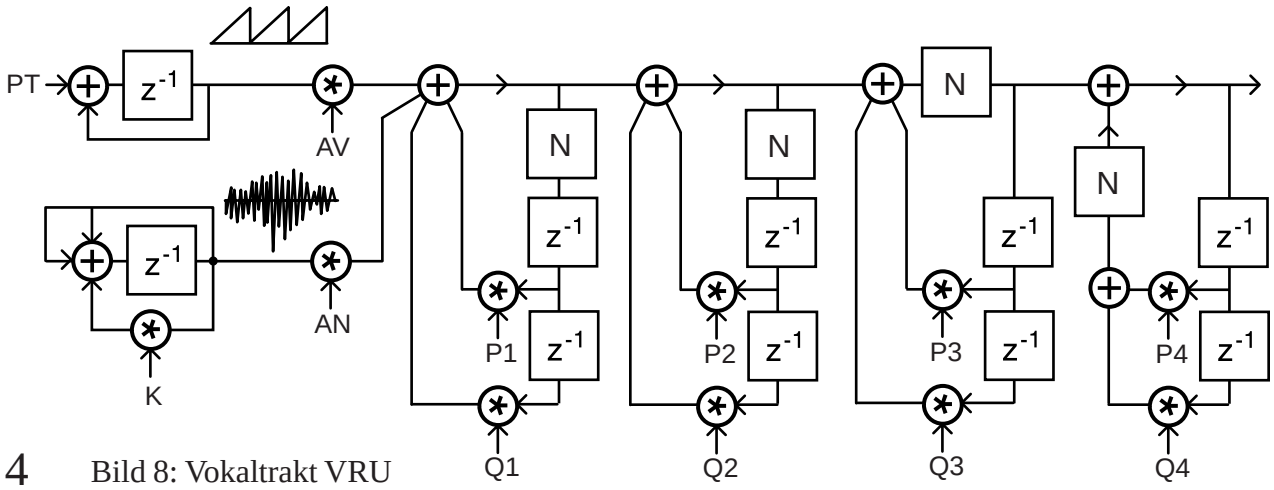


Bild 8: Vokaltrakt VRU

$x_{n+1} = (a * x_n + b) \bmod m$

a = E51D
b = 3619

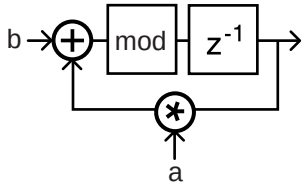


Bild 11: LCG-Generator

Pitch

Wenn man einen festen Wert PI integriert (Bild 8) ergibt sich durch Moduloüberlauf ein Rampengenerator. Die Größe von PI bestimmt die Pitchfrequenz die damit leicht einstellbar ist. Dieses aufwandsarme Verfahren wurde für das IC beibehalten. Der Oberwellengehalt ist etwas geringer als erwünscht. In [6] wird deshalb Preemphase bei der Analyse der Sprachdaten empfohlen.

Manche LPC-Synthesizer verwenden gesampelte Pitch Impulse aus Tabellen. Der geringe Mehraufwand in der Anregung senkt die Anforderungen an die folgenen Filter und ist damit sehr empfehlenswert.

Rauschen

Simpelster digitaler Generator wäre ein LFSR wie es zeitweise von Votrax propagiert wurde. Wegen der Filter ist hier aber ein schneller Multiplizierer verfügbar. Als Generator wurde deshalb in der VRU ein LCG [7] gewählt (Bild 8). Der qualitativ auch besser verteilte Werte als ein LFSR liefert. Beide Verfahren erzeugen zwar nicht gaussche sondern Gleichverteilung. Aber das folgende Vokaltraktfilter behebt das weitgehend.

Die genaue Ausführung des

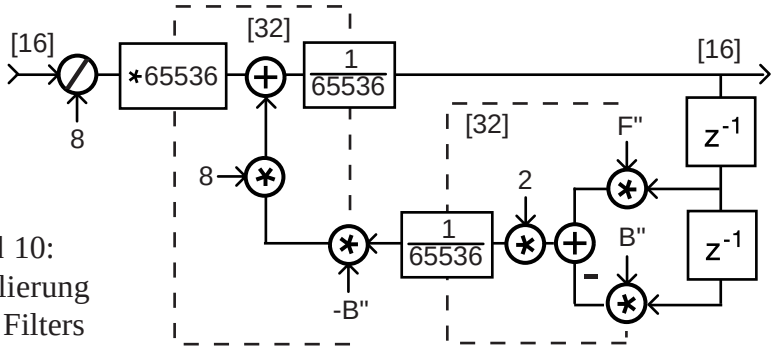


Bild 10: Skalierung des Filters

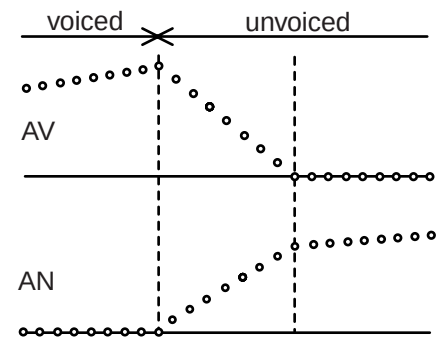
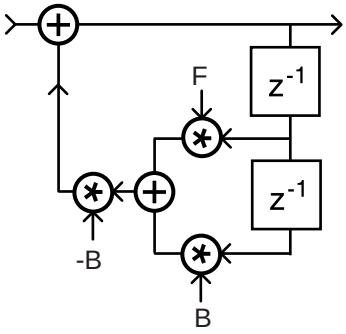


Bild 12: Interpolation

LCG im IC ist nicht bekannt. Für den Simulator wurde eine übliche 16 Bit Variante gewählt (Bild 11). Nach Reset die Initialisierung auf einen Wert ungleich Null nicht vergessen.

Interpolation

Die Werte für AMPL, BWx und FMx werden innerhalb eines Frames mit 7 Stützstelle linear interpoliert. Hier am Beispiel AV und AN gezeigt (Bild 12) von denen jeweils einer auf den linearisierten Wert von AMPL erhält während der andere Wert auf Null gesetzt wird. Da die Dauer des Frames abhängig von FD von 8 - 64msec variiert (Tabelle 3), ändert sich der Interpolationsschritt 1 - 4msec.



$$F = 2 * \cos (2\pi \frac{f_m}{f_s})$$

$$B = e^{-(\pi \frac{b}{f_s})}$$

Bild 9: verbessertes Filter

Tabelle 5: Skalierung FM

Hz	F=	F*32767/2
150	1.986	32540
162	1.983	32502
174	1.981	32462
188	1.978	32410
202	1.974	32355
217	1.971	32292
233	1.966	32220
250	1.961	32137
267	1.956	32049
286	1.949	31944
305	1.942	31831
325	1.935	31705
346	1.926	31565
368	1.917	31408
391	1.906	31234
415	1.894	31042
440	1.881	30830
466	1.867	30597
494	1.851	30332
523	1.833	30041
554	1.813	29714
587	1.791	29346
622	1.766	28934
659	1.738	28475
698	1.706	27965
740	1.671	27387
784	1.632	26749
830	1.589	26048
880	1.541	25247
932	1.487	24374
988	1.427	23387
1047	1.361	22299
1110	1.286	21084
1179	1.202	19695
1254	1.105	18119
1337	0.995	16302
1428	0.868	14230
1528	0.724	11871
1639	0.559	9166
1761	0.373	6115
1897	0.161	2648
2047	-0.073	-1209
2214	-0.334	-5481
2400	-0.618	-10126
2609	-0.920	-15082
2842	-1.228	-20124
3105	-1.525	-25000
3400	-1.782	-29196

Tabelle 6: Skalierung BW

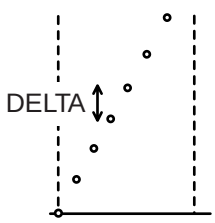
Hz	B=	*32767/2
726	0.752	12319
309	0.886	14511
125	0.952	15599
50	0.981	16065

Auch die Pitch-Frequenz wird entsprechend PI im Frame schrittweise verändert, hier ist aber der Takt fest 8msec.

Letztlich bedingt die Interpolation erstens, daß man einen Frame erst ausgeben kann nachdem man bereits den nächsten Frame aus dem Speicher geholt hat. Zweitens muß man die Daten beider Frames aus dem komprimierten Format in lineare +/-15 Bit Zahlen wandeln. Dann kann man für die interpolierten Werte eine +/-15 Bit Konstante (DELTA) berechnen die in jedem Interpolationsschritt addiert wird (Bild 13).

Letztlich hat man also 3 Datensätze (Tabelle 7): den gerade ausgegebenen Wert `FRAME"`, den addierten Interpolationswert `DELTA` und die Daten des nächsten Frame `FRAME""`.

Bild 13: Interpolation über Addition von Konstante



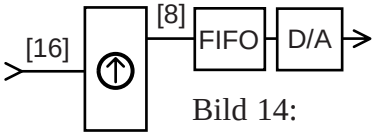


Bild 14: Interpolation und FIFO

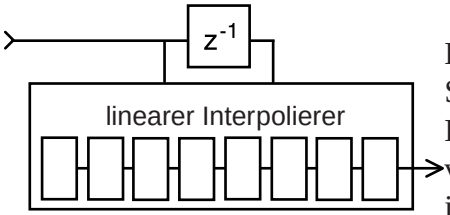
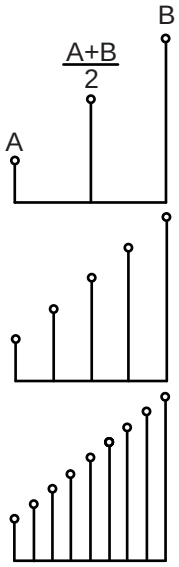


Bild 15: Interpolator für Bild 14

Tabelle 7: Datensätze

*FRAME": *	*DELTA:*	*FRAME"":*
BW1"	D-BW1	BW1""
BW2"	D-BW2	BW2""
BW3"	D-BW3	BW3""
BW4"	D-BW4	BW4""
FM1"	D-FM1	FM1""
FM2"	D-FM2	FM2""
FM3"	D-FM3	FM3""
FM4"	0	FM4""
AP"	D-AP	AP""
AN"	D-AN	AN""
PI"		PI""
FD"		FD""
PITCH"	D-PI"	

Bild 16: Schrittweise Berechnung von 7 linear interpolierten Stützstellen



D/A

Vor dem D/A Wandler (Bild 14) ist in jedem Fall eine FIFO nötig, die die variable Rechenzeit des Controllers kompensiert. Da der Wandler nur 8 Bit Auflösung hat wird nur die obere Hälfte des Datenworts weiterverarbeitet. Es wird angenommen, daß es sich um ein R2R-Netzwerk handelt, auf die Nachbildung der PWM-Schaltung wird verzichtet.

Im MEA8000 erhöht ein linearer Interpolator die Samplerate von 8kHz auf 64kHz (Bild 15). Wenn man das Signal für den D/A auf vorzeichenloses Format gewandelt hat bietet sich das simple Schema an das ohne Multiplizierer die 7 Stützwerte erzeugt (Bild 16). Die Grundroutine `(A+B)/2` programmiert man vorzugsweise in Assembler weil man dann das Carrybit als Zwischenspeicher für die Addition effizient nutzen kann.

[7] emb (13) 2pol Allpol Filter
[8] emb (4) PRNG Lineare Kongruenz