

embedded



Voransichts- Version

Für Bezug des Originals
siehe FAQ auf
www.embeddedFORTH.de

Rafael Deliano
Steinbergstr.37
82110 Germering
Tel 089/8418317

j_r_d@t-online.de

V1.0 (pdf) : 28. April 02
V1.1 (pdf) : 30. April 04
V1.2 (pdf) : 14. Jan 07

READ . ME

- 1 Inhalt, Impressum
- 2 Zyklische Codes Teil 2
- 5 Multiplizierer, Graycode
- 6 Floatingpoint-
coprozessor
- 12 „Idealer“ FIR Tiefpaß
- 14 1 Bit FIR in Hardware
- 16 1 Bit FIR in Software
- 17 TTY Teil 1

Die Zeitschrift ist nun seit 12 Monaten im www verfügbar. Dank google war die Nachfrage mit typisch 35 Downloads pro Monat je Heft überraschend gut.

Aufgrund des positiven Zuspruchs von Seiten der Leser habe ich kurzfristig aus dem Material das sich inzwischen angesammelt hat eine weitere Ausgabe zusammengestellt, obwohl die Zahl der behandelten Themen nicht so reichhaltig ist wie ich eigentlich beabsichtigte. Aus den nicht fertiggewordenen Artikeln wird jedoch dieses Jahr noch ein weiteres Heft folgen.

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

Zyklische Codes Teil 2

Weitere Codes und Decoder für zyklische Codes, speziell Error Trapping, eine effiziente Variante des Meggitt-Decoders. Zuvor jedoch ein kurzer Nachtrag.

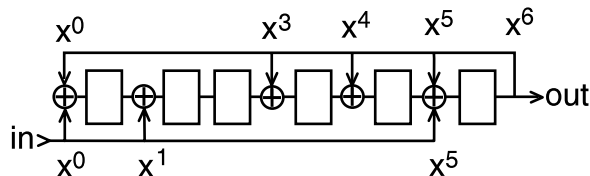
Die Darstellung im letzten Heft, daß sowohl der Coder als auch der Decoder Division verwenden ist wenig intuitiv. Denn man sollte erwarten, daß der Coder eine Multiplikation verwendet.

In Bild 1 sei deshalb gezeigt wie im Coder die Prüfsumme durch Multiplikation mit der unbekanntem Konstante G' erzeugt wird. Damit man aus Prüfsumme und Daten ein gemeinsames Nachrichtenwort „message“ bilden kann, muß das Datenwort hochgeschoben werden, was der Multiplikation mit einer Konstante $g^{(n-k)}$ entspricht. Im Decoder wird das Nachrichtenwort durch das Generatorpolynom G , einer bekannten Konstante, dividiert. Bei fehlerloser Übertragung ist das Ergebnis Null. Damit kann man die Konstante G' aus der 1. Zeile bestimmen. Das störende Minuszeichen vor G' soll später mal in einem Galois-Feld entsorgt werden.

Das Verfahren führt zu gleichzeitiger Multiplikation und Division im Encoder wie sie in Bild 2 nochmal für den Hamming (7,4) gezeigt ist. In Bild 3 sind weitere Beispiele für diese Schieberegistergrundschaltung gezeigt.

Bild 3: Beispiele für gleichzeitige Multiplikation & Division

$$\frac{g(x)}{h(x)} = \frac{x^5 + x + 1}{x^6 + x^5 + x^4 + x^3 + 1}$$



$$\frac{g(x)}{h(x)} = \frac{x^{10} + x^9 + x^5 + 1}{x^6 + x^5 + x^4 + x^3 + 1}$$

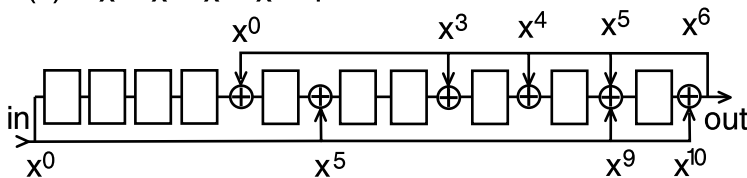
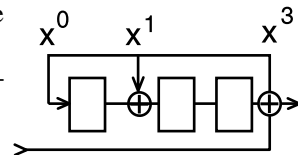


Bild 2: Encoder (7,4)

$$\frac{g(x)}{h(x)} = \frac{x^3 + x + 1}{x^3}$$



Sie ist in Hardware sicher nützlich und findet sich häufig für gekürzte zyklische Codes („shortened cyclic codes“). In Software weniger vorteilhaft. Hier seien deshalb gekürzte Codes durch Wegwerfen und Einfügen von Datenbits verarbeitet.

Burst Error Correcting Codes

Die bekanntesten Bündelfehlercodes dürften Fire-Codes sein wie sie in den 70er Jahren in Harddisks und Magnetbändern verwendet wurden. Daneben findet man in [1] jedoch eine Liste von zum Teil recht kurzen Codes die es erlauben mehrere aufeinanderfolgende gekippte Bits zu korrigieren.

Der (15, 9) war als gekürzter (14, 8) für das ursprüngliche Powerline-Modem Protokoll von EHS vorgesehen. Er kann drei aufeinanderfolgende zerstörte Bits korrigieren. Es ist also wohlgermerkt kein Code der

$$Z' = \text{Prüfsumme} - \text{Prüfsumme} + \text{Daten} \cdot g^{(n-k)}$$

Bild 1: Rechenschaltung des Coder (Multiplikation) und des Decoder (Division)

... nicht? Das kommt, wenn bei einem zufälligen Fehler die Prüfsumme zufällig im Datenwort ankommt sein. Man kann jedoch Mehrbitfehler nachträglich durch Treuefehler mit Codes die Fehlerkorrektur besitzen vermeiden. In [9] findet man die Beschreibung der Fire-Codes und die Eigenschaften der Fire-Codes. In [10] wird die Beschreibung der Fire-Codes in FÜRTEL dargestellt. In [11] wird die Beschreibung der Fire-Codes dargestellt. In [12] wird die Beschreibung der Fire-Codes dargestellt.

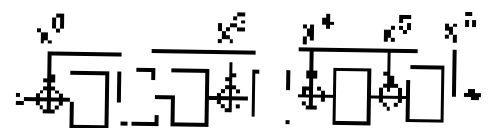


Bild 4: Division mit Polynom des Bündelfehlercodes (15,9)

... Nach dem Codes Symbolgenerations kann man die Fehlerkorrektur durchzuführen (siehe [1]).
 ... Für die Fehlerkorrektur ist die Fehlerkorrektur im Ende der Tabelle in den Datenwortesymbolen werden. Zyklische Codes verwenden zyklische Fehler. Zwei sind es nicht? In Tabelle sondern in 4. und manche Fehler werden nur wenn man die Fehler in anderer Reihenfolge von Anfang wird das Minus von MSB (Most Significant Bit) ist egal. Das liegt daran, daß der Datenwort (5 bit) über ein Minus (6 bit) und liegt. Da man in der Praxis nur die Fehler im Datenwort korrigieren will, werden diese 1. und 2. Stellen oft nur gelöscht auf die Datenbits gestellt. (siehe [9] Zeilen)

Der Encoder und Decoder benötigen jeweils 20 Bit für die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162.

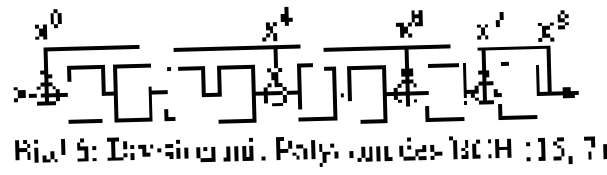
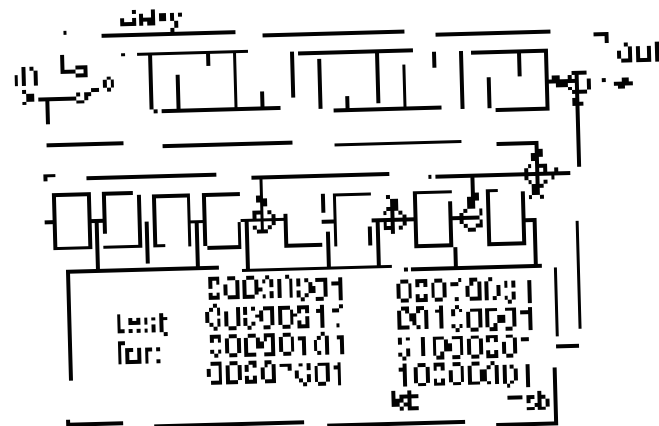


Bild 7: Error Trapping Decoder für (15, 7)



Error Trapping

Die Anwendung des Syndroms ist in 5 Bit durch BCH (15, 7) möglich. In Subtrahierern Form von Spaltenmatrix. Die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162.

Das Trapping ist eine Variante des Mergen-Prozesses die 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162.

Das ist die Tabelle in der Tabelle 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162.

Der Code im Beispiel im BCH (15, 7) wird durch die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162.

Die Tabelle 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162.

Tabelle 1: Ablauf einer Syndrommodifikation

nach dem 10. und 7. Fehlerblock	0010010000000000
nach dem 10. und 15. Fehlerblock	0010000100010000
nach dem 10. und 15. Fehlerblock	0000000010000000
1	0000000010000000
2	0000000010000000
3	0000000010000000
4	0000000010000000
5	0000000010000000
6	0000000010000000
7	0000000010000000
8	0000000010000000
9	0000000010000000
10	0000000010000000
11	0000000010000000
12	0000000010000000
13	0000000010000000
14	0000000010000000
15	0000000010000000
16	0000000010000000

Tabelle 3: wie Bild 11 aber mit Syndrommodifikation

0	0000000010000000	1
1	0000000010000000	2
2	0000000010000000	3
3	0000000010000000	4
4	0000000010000000	5

Verfahren

Das Verfahren ist ein zweifacher Prozess. In der ersten Phase wird die Tabelle 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162. Es gibt sich noch die Werte 20 (11) + 10 als die Werte für die Leistung 1.162.

11. Ein Prozess, Error Control Coding, Thomas 1987

Floatingpoint-Coprozessor für Controller

In Steuerungen verwendet man vorzugsweise 8 oder 16 Bit Controller. Leider ist für sie oft keine Gleitkomma-Software („float“) verfügbar. Die publizierten, oft recht alten, Programmbibliotheken leiden meist unter geringer Genauigkeit. Floats mit hoher Auflösung in Software auf den kleinen CPUs benötigen hingegen unangenehm lange Rechenzeit und sind deshalb auch nicht praktikabel.

Soweit Geräte nur in minimalen Stückzahlen gefertigt werden und Stromverbrauch nicht allzu kritisch ist, ist die Verwendung eines Coprozessors (FPU) eine wirksame Lösung. Dieser rechnet in voller IEEE754-Auflösung. Und er tut das schnell: einfache Befehle wie Multiplikation in 4usec und Sinus in 40usec.

FPU's

Intel begann 1975 mit der Entwicklung des 8087 und brachte ihn 1980 auf den Markt. Das Normungskomitee für den IEEE754 Standard tagte hingegen von 1977 bis 1985. Intels 80287 entsprach auch erst dem provisorischen Standard von 1982. Ferner berechnen diese beiden ersten FPU's noch keinen Sinus. Erst der 80387 basiert auf dem endgültigen IEEE-Standard. Es gab zwei 80387-Clones, den Cyrix 83D87 und den IIT NP-3C87. Ab 80486 ist bei Intel die FPU in den Prozessor integriert.

Motorola kündigte 1982 den 68881 an, aber wartete vorsichtshalber mit der Fertigung bis der IEEE-Standard verabschiedet war. Der geringfügig verbesserte 68882 kam bald darauf 1987 auf den Markt. 1990 verkündete Motorola vollmundig, der 68040 brauche keinen Coprozessor, da er so schnell sei, daß er alles in Software könne. Der Markt war für Motorola zu klein um die Entwicklung einer FPU sinnvoll erscheinen zu lassen.

Es gab noch weitere Coprozessoren von Weitek oder auch für Transputer, aber die Verbreitung war minimal. Heute sind alle diese IC's

obsolet. Man ist damit auf Typen angewiesen die damals in Massen produziert wurden und heute aus ausgeschlachteten Leiterplatten und Überbeständen billig auf den Markt kommen. Diese Welle wird vielleicht in den nächsten Jahren zurückgehen und die Verfügbarkeit wird dann nicht mehr gegeben sein. Derzeit ist aber der Preis niedrig, man bekommt gebrauchte Teile für ca. 20DM/Stück.

MC68882

Der 80387 hat eine dem 68881 entsprechende Rechenleistung und von ihm sind sicher mehr verkauft wurden. Die Motorola-Typen sind aber besonders leicht an fremde CPUs anzuschließen, da ihr Businterface auf 8, 16 und 32 Bit Datenbus konfiguriert werden kann. Dabei hat der 68882 ein etwas einfacheres Timing und ist handelsüblicher. Man erhält ihn typisch als Keramik PGA68 (Code „RC“) (Bild 1) oder seltener als PLCC68 (Code „FN“) wofür keine Pinbelegung aufzutreiben war. Die ursprünglichen Geschwindigkeitsselektionen sind 16, 20, 25 und 33 MHz.

Schaltung

Den Takt wird man durch einen lokalen Oszillator erzeugen. In Bild 2 ist eine diskrete Schaltung gezeigt. Bei den hohen Frequenzen sind Verstärkung und Phasendrehung des 74HCU04 schon recht kritisch. Zudem kriegt man 20 MHz zwar meist noch als Grundwellenquarz, aber 32MHz kann schon ein Oberwellenquarz sein, was eine

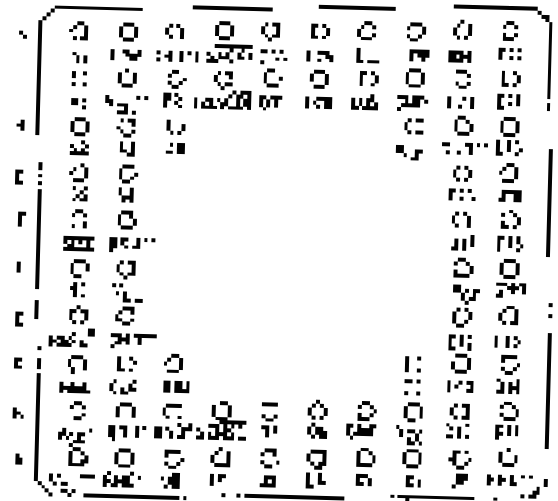


Bild 1: Blick auf die Pins des PGA.

Markierung ist Ecke A1 andere Schaltung erfordert. Wer also nicht gerne mit solchen Schaltungen kämpft, sollte oberhalb 20 MHz eine fertige Dose nehmen, die ja heute schon recht billig sind.

Die FPU ist eine Mischung aus CMOS und NMOS und damit nicht voll statisch. Sie kann nur in einem engen Taktbereich betrieben werden (Tabelle 1). 16MHz passen zwar immer. Aber wegen Zugriffszeit vom Controller will man hohen Takt. Für Stromverbrauch wäre langsamer geringfügig (NMOS) besser.

Tabelle 1: Taktfrequenzen in MHz

nom	min	max
16,67	8	- 16,67
20	12,5	- 20
25	12,5	- 25
33,33	16,7	- 33,33

Stromaufnahme ist etwa 150mA. Es empfehlen sich mehrere Sätze von 10uF Tantalperlen, 100nF und 10nF nahe am IC. Bei Fädeldrahtaufbauten sollten die vielen Stromversorgungspins durch mehrere Leitungen vermascht verdrahtet werden. Eigentlich erwartet so ein IC eine Multilayerleiterplatte.

Recht einfach ist die Busbeschaltung, die hier für 8 Bit Datenbus gezeigt ist. Jeder Pin des Controllerdatenbus ist an der FPU an 4 Pins anzuschließen.

Die FPU belegt 32 Bytes memorymapped im Speicher (Tabelle

Tabella 4: Operazioni

Operazioni			
Op	OpCode	OpLen	OpName
01	0000	15	ADD
02	0001	16	ADD
03	0002	17	ADD
04	0003	18	ADD
05	0004	19	ADD
06	0005	20	ADD
07	0006	21	ADD
08	0007	22	ADD
09	0008	23	ADD
0A	0009	24	ADD
0B	000A	25	ADD
0C	000B	26	ADD
0D	000C	27	ADD
0E	000D	28	ADD
0F	000E	29	ADD
10	000F	30	ADD
11	0010	31	ADD
12	0011	32	ADD
13	0012	33	ADD
14	0013	34	ADD
15	0014	35	ADD

Man kann die Befehle auch in 3 Gruppen unterteilen.

Es gibt unimodale und Arithmetische Befehle für die Arithmetischen Operationen. In diesen Befehlen sind die Quelle und die Ziel angegeben. Die Befehle sind:

- ADD: Addition
- MUL: Multiplikation
- DIV: Division
- SUB: Subtraktion
- MOD: Modulo
- AND: AND
- OR: OR
- XOR: XOR
- NOT: NOT
- SHL: Shift Left
- SHR: Shift Right
- ROL: Rotate Left
- ROR: Rotate Right

Schliefend gibt es für Datenbewegung die Befehle mit Quelle und Ziel. Sowas kann man verwenden, um Daten von einem Register zum anderen zu verschieben. Die Befehle sind:

Operanden

Die Operanden können von 8 bis 32 Bit (1 bis 4 Bytes) lang sein. Das Format entspricht immer dem Byte von der niedrigsten Lage (Tabella 5). Jeder Operand hat eine

Tabella 5: Operatoren

Oper	OperLen	OperName
000	8	Byte
001	16	Word
010	32	DWord
011	64	QWord
100	8	Byte
101	16	Word
110	32	DWord
111	64	QWord

Man kann die Befehle auch in 3 Gruppen unterteilen.

Es gibt unimodale und Arithmetische Befehle für die Arithmetischen Operationen. In diesen Befehlen sind die Quelle und die Ziel angegeben. Die Befehle sind:

- ADD: Addition
- MUL: Multiplikation
- DIV: Division
- SUB: Subtraktion
- MOD: Modulo
- AND: AND
- OR: OR
- XOR: XOR
- NOT: NOT
- SHL: Shift Left
- SHR: Shift Right
- ROL: Rotate Left
- ROR: Rotate Right

Tabella 6: Konstanten im CPU-ROM

Op	OpCode	OpLen	OpName
00	0000	15	ADD
01	0001	16	ADD
02	0002	17	ADD
03	0003	18	ADD
04	0004	19	ADD
05	0005	20	ADD
06	0006	21	ADD
07	0007	22	ADD
08	0008	23	ADD
09	0009	24	ADD
0A	000A	25	ADD
0B	000B	26	ADD
0C	000C	27	ADD
0D	000D	28	ADD
0E	000E	29	ADD
0F	000F	30	ADD
10	0010	31	ADD
11	0011	32	ADD
12	0012	33	ADD
13	0013	34	ADD
14	0014	35	ADD

Register für Register. Die Register sind:

Register	RegisterLen	RegisterName
000	8	Byte
001	16	Word
010	32	DWord
011	64	QWord
100	8	Byte
101	16	Word
110	32	DWord
111	64	QWord

Control Register

Die Befehle in Tabella 7 sind für die Control Register. Die Befehle sind:

- CLR: Clear
- SET: Set
- AND: AND
- OR: OR
- XOR: XOR
- NOT: NOT
- SHL: Shift Left
- SHR: Shift Right
- ROL: Rotate Left
- ROR: Rotate Right

Das Control Register wird im CPU-ROM initialisiert. Die Befehle sind:

Die Befehle sind:

Tabelle 10: CS-Register

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS: Code Segment Register																CS: Code Segment Register																

CS: Code Segment Register
 Bit 31-0: CS: Code Segment Register
 Bit 31-0: CS: Code Segment Register

CR0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR0: Control Register																CR0: Control Register															

Zugriff auf diese Adressen geschieht nur auf unkonventionelle Weise. Die Adressen sind in 32-Bit-Form (Tabelle 27) damit ist hier eine 32-Bit-Schrittweite in einer Meladirektive möglich. Weiterhin ist die Register-Form und die des binären dargestellt. Die erste Byte immer der Highbyte (H) die zweite Byte die Lowbyte (L). Einige Register kann nur mit einer (L) oder schreiben (H) oder beiden (HL). An jedem Register in dieser Anwendung benutzt werden. Register kann auch über der Adressen von 10 bis 16 Byte reduziert, wenn man den Decoder verknüpfte.

CR2: Base Control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR2: Base Control																CR2: Base Control															

CR3: Base Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR3: Base Register																CR3: Base Register															

CR4: Feature Status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR4: Feature Status																CR4: Feature Status															

CR8: Condition Code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR8: Condition Code																CR8: Condition Code															

Franken

Nach dem Schreiben des Operands ins Zielwert Register CR8 muß der Controller das Register-Register CR8 lesen. Von dem wird einem registriert, ob die CR8 nach beschriebener, oder man wird aufgefunden der Operationen zu schreiben falls ein Fehler vorgetreten ist. Dann muß man die Register CR8 lesen, damit die CR8 mit der Antwort erfolgt. Wenn die CR8 fertig ist, dann man durch Kontrolle CR8 lesen feststellen.

FPU-Assembler

Auf dem ersten Level wird man für den 68882 einen Floating-Assembler schreiben wie für einer Controller. Selbst wenn die Programmierung von Anwendungen in FPU-Assembler ist, ist es möglich die Entwicklung der FPU und schließlich zum Schreiben von einer Assembler kommen.

Tabelle 1 und Tabelle 2 enthält die Liste der arithmetischen Funktionen. Ein Kontrolle mit dem Assembler des Computers zu kommen ist es hier die Operationen, die die CR8 wie oben beschrieben. Die meisten Funktionen über den zwei Operanden wird Stück, um die Quelle und den die Zielregister. Somit es sich um FPU-Register immer, die Ziel-Register Nummer.

$$R1 \leftarrow R2 + R3 \quad R4 \leftarrow R5 * R6$$

$$R7 \leftarrow R8 * R9 \quad R10 \leftarrow R11 * R12$$

Nur bei R0-R7 kann die Quelle man mit Konstanten R0-R7 sein.

$$\text{and } R1, R2, R3 \quad \text{or } R4, R5, R6$$

Somit versuchen Operationen die Quelle möglichst um zwei Operanden angegeben sind. Anders als bei ihrem Floating-Assembler liegt es im Design allerdings wegen der Länge nicht auf dem FPU-1 Stück, sondern müssen werden. In der 12-Bit Variable (R0-R7) abgelegt werden sein. Adressierung für die Daten. Daten sind in Tabelle 1-30. Ziel-Register sind in der Tabelle 1-30. Daten sind in der Tabelle 1-30. Daten sind in der Tabelle 1-30.

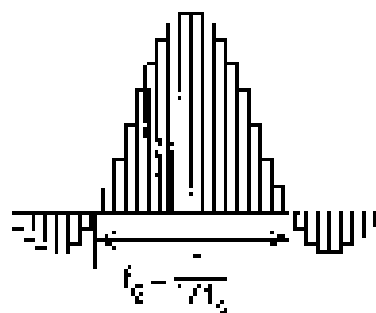


Bild 7: Grenzfrequenz

benutzt, um die ursprüngliche Form. Sie wird durch periodische Multiplikation mit der Fensterfunktion in die Koeffizienten umgewandelt. Diese Fensterfunktion ist hier ein Rechteckfenster, es wäre für alle nur Sinus und Cosinus die Köpfe überstrichen. Ein Fenster lässt sich als Filter pro Formel auch negative Frequenzen. Folgende können entgegengesetzt die bekannte Zeitgleichheit verwendet man die Variante im Frequenzbereich mit einem zentralen Maximum. Die Zeitgleichheit ergibt sich durch die Gleichzeitigkeit. Folgende sind Gleichzeitigkeit Fenster als z.B. Dämpfung Grenzfrequenz.

Filterband

Die Filterfrequenz der digitalen Frequenzgang (Bild 8) zeigt ein reines Frequenzgang (Bild 8) von Wert mit Verstärkung 0,5, nicht der 0,3 Punkte wie bei anderen Filtern gegeben.

Die Grenzfrequenz ergibt sich aus dem Kehrwert eines vollen Systems der ersten 60 Punkte in der Impulsantwort (Bild 9). Im gewöhnlichen Hebel. Dieser aus 17 Schritte der Impulsantwort liegt ursprünglich in der ersten Hälfte von (Bild 8). Der Wert bei 0 ist nicht bei der Rechnung auf 1 gesetzt, sondern die Division durch den ersten Wert. Durch Multiplikation der Zeitgleichheit für die gewöhnlichen Koeffizienten für das Filter.

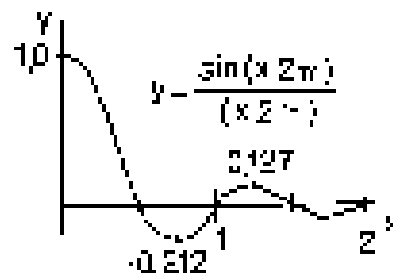


Bild 8: Sinus-Häufung-Funktion

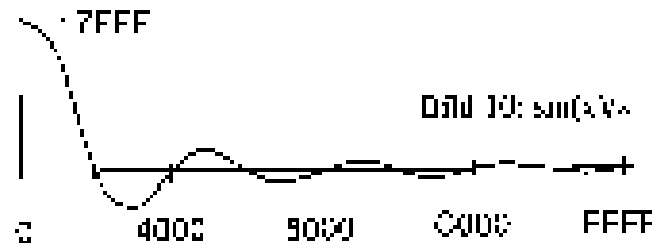


Bild 9: Sinus-Häufung-Fensterfunktion

Sowohl man mit der niedrigen Genauigkeit leben kann, zum man die benötigten Berechnungen in Festkomma-Direkt auf einem Digitalrechner auszuführen. Mit dem Wert, hat man mit diesen auch die Simulation durchführen kann. Ausgangspunkt für das hier verwendete Programm (using STK in Anhang) ist ein Simulationsprogramm von einem Bild 9. Es ist ein 20000 Punkte bestehende z-Block für den es sich auf 4 Ziffern beschränkt und wie in Bild 10 gegeben.

Die Resulte (Bild 11) (Bild 11) zeigt die Liste der Koeffizienten der 16 Bit Binär-Komplettzahl. Als Vergleich ist die Ordnung des Filters $0,5 \cdot 200 = 100$ gegeben. Hier werden nur gerade Punkte für Ordnung 8 bis 64 berücksichtigt. So ist ein Grenzfrequenz bei dem man nicht wiederholt von 0 bis 100000000. Ein 32 Tage Filter der kann bei Bild 7 berechnet man mit der Gleichheit.

Bild 10: Sinus-Häufung-Fensterfunktion

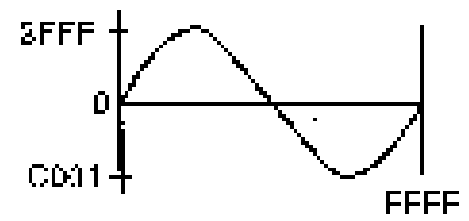


Bild 9: Sinus-Häufung-Fensterfunktion

Hanning-Fenster

Nach der Anwendung des Hochpassfilters erhält man eine Kosinusfunktion von 32 Werten. Dieses sind man auch ein gleichzeitiges Fenster wie ein modifiziertes. Besonders bekannt ist das Hanningfenster (Bild 12). Die entsprechende Impulsantwort ist wie in Bild 12 gezeigt. Skalieren diese Fenster mit dem Wert 0,5 und die Dämpfung im Frequenzbereich durch den Frequenzgang.

In den folgenden Berechnungen wird das Hanningfenster nicht verwendet.

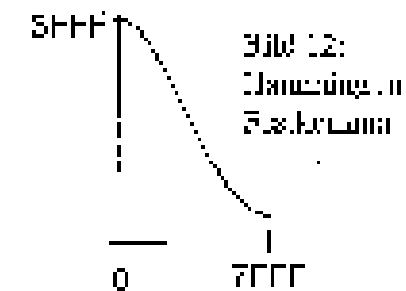


Bild 12: Hanning-Fensterfunktion

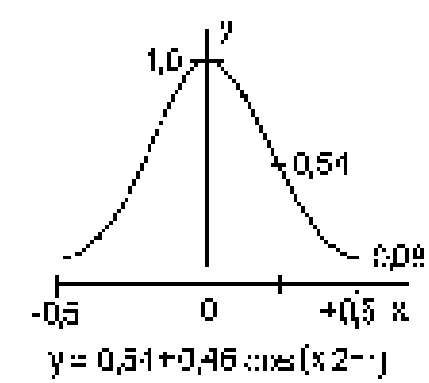


Bild 11: Hanning-Fensterfunktion

Man hat also zu übermitteln, was emp-
fängerlich ein Programm aus alle Variablen
darüber und den jeweiligen Gesamtfehler
beschreibt (Lösung RPT im Anhang).

Im Programm sind vier 121 Widerstände
von 1k Ω bis 10k Ω in 10 Ω Schritten
Leistung in 100 μ W angegeben. Unter der Annahme
einer konstanten Spannung, die hier je
gegeben ist, kann man diese Werte auch als
Strom auflesen. In Zeile 10 sind die
sich die Bezeichnungen.

Nachdem man sich den gültigen Wert in
die Kartikantenmenge als Struktur gegeben
hat, sind geeignete Skalierung vorgenommen
wurde. Können sich die Werte um 20 bis 200
Widerstände dann in diese Liste festhalten
werden, bzw. ergeben sich typischerweise
einmal ein bestimmter Wert, der zu groß ist
und immer die 100 ist. Es sind also jeweils
die Werte mit dem kleinsten Fehler bestimmt
und gewählt werden. Gemeint ist die
Summe aller 12 Einzelfehler. Für die
12 Rang der Standardabweichung. Diese
werden in der Liste Widerstände stärke,
bezeichnet als Bezeichnung. Das entspricht der
Geometrie der bei Ihnen.

NA Werten zwischen 10 μ g bis die
in die großen angeführten zusammen
sind diesen Prozess gegeben sein.

In der Tabelle ist die Liste der
Funktion. Somit sind die Variablen C, 254
die Gesamtfehler analysieren und den
dies durch die Werte die Widerstände der
Liste geben in Lösung 1.

Bad 2 zeigt den Abwärtstrend unterhalb
der Grundfrequenz der Leistung. Die
Vergleichspunkt in Bereich der Grenzfrequenz
ergeben sich bei 10 μ g bis zu einem
(Bild 6). Durch den Fall der Widerstände
beschreibt sich die Leistung auf etwa 10 μ g
Bilder.

1) Kalkulation der Energie
aus statistischen Zusammenhänge
Bild 1: 3-9%

Listung 1: Eingaben & Auswerte

1000 10 100-10 10 calculate

2020	1	prime	number	of	calculations	0	171
01	1916	01	1916	02	1916	03	2000
04	1947	16	2004	07	2007	11	2015
08	2005	18	2022	07	2105	09	2600
04	2701	11	2707	11	2828	12	3012
10	2477	17	2918	15	2901	19	3106

2	CALL	1	number	of	calculations	0	171
01	- 1072	10000	01a	-R	+ 100	1a:	-
01	- 6872	1000	01b	-R	- 40	1b:	- 107
02	6872	1000	01b	-R	- 40	+F:	- 279
03	1050	5700	01a	-L	- 10	1a:	- 252
04	6901	5611	01a	-F:	- 251	L:	- 1
05	1091	6101	01b	-E:	- 201	E:	+ 23
06	1076	12000	01a	-E:	+ 50	-R:	+ 10
07	7	200000	01b	-R	+ 0	L:	+ 1
08	1420	1100	01b	-E:	+ 10	+R	+ 14
09	+ 9705	4100	01a	-R	+ 15	1a:	- 67
08	-14412	2000	01a	-E:	- 40	+F:	- 21
07	-19004	2000	01b	-E:	- 25	1b:	- 901
01	10001	1000	01b	-E:	- 100	-F:	- 28
02	10001	1000	01a	-L:	- 40	-E:	+ 106
0F	+1000	1000	01b	-F:	+ 250	E:	+ 110
04	10000	1000	01a	-E:	+ 250	-R:	+ 10
05	+1000	1000	01b	-R	+ 100	E:	+ 1
11	10100	1000	01b	-E:	+ 40	+R	- 610
12	-20204	1000	01a	-R	- 40	1a:	- 11
11	-24321	1000	01b	-F:	- 110	1b:	- 28
14	-19004	1000	01b	-E:	- 20	+F:	- 981
15	-10019	2000	01a	-F:	- 500	1a:	- 1
16	+ 3179	4000	01b	-E:	- 30	-E:	+ 101
17	+ 4009	9100	01a	-E:	+ 100	-E:	+ 10
18	7	200000	01b	-R	+ 0	L:	+ 1
19	2000	1000	01b	-E:	+ 10	+R	- 72
1A	- 5001	2000	01a	-E	+ 100	1a:	- 21
1B	- 6001	2000	01b	-E:	- 20	+F:	- 31
1C	1050	5700	01a	-L:	- 10	+E:	- 252
1D	- 6310	6200	01b	-F:	- 30	L:	- 279
1E	4072	1000	01a	-L:	- 40	-R:	+ 101
1F	- 1002	10000	01b	-E:	+ 100	E:	+ 1



Bild 5: Eingang Rechtecksignal
64 Bit, high 64 Bit low



Bild 6: Eingang Rechtecksignal
16 Bit, high 16 Bit low

Schaltung Bild 1



Schaltung Bild 2

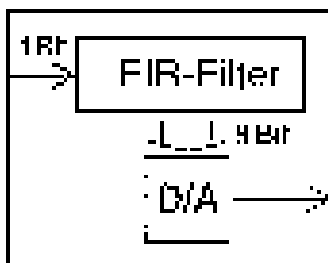


Bild 7: Eingang Rechtecksignal
4 Bit, high 4 Bit low

1 Bit FIR Filter auf Controller

Die Implementierung in Software ist recht effizient, da keine Multiplikatoren benötigt werden. Auch für Entwicklung der Hardware vorzugsweise mit Schieberegistern ist eine Vorteil in Software mit referenzierten Faktoren günstiger als Überbrücken von Widerständen.

Controller:



Die Daten des Filters werden dabei über einen Komparator (oder das Gegenstück der D/A-Wandlung ausgegeben) (Bild 1). Allerdings können die Daten zusätzlich auch im Controller erzeugt werden.

Realtime

Bild 1: Rückkopplung

Das Ringregler (Bild 2) wird durch 4 kostengünstige KOP-Intensivmax 12 Bit Register (2) erzeugt. Abhängig davon ob die Bit im Schieberegister gesetzt ist oder nicht werden die 16 Bit Koeffizienten zu 8 Bit aktiv, oder nicht, sie sind im Pre-Komplement eingestellt, so daß auch negative Konstanten durch Addition verarbeitet werden können. Die Abhängigkeit der Koeffizienten nach positive summen entstehen können ist abschließend die Addition eines Vorzeichens über die des Vorzeichenbit eines zweizeilenwertes Kennen wandeln. Die Summe wird dann durch zwei 8-Bit Register addiert, und die unteren 8 Bit an dem D/A-Wandler ausgegeben.

Hier wird die die 37421 ein 16-Bitbit 6500 mit 20480 verwendet über einen 8-Bit-D/A-Wandler eingebaut hat. Beachtung eines Samples durch ca. 90µs. Es wird eine Dynamik von über 35dB erreicht, was etwas besser als die typische Werte mit Schieberegistern und Widerständen ist (vgl. Bild 3).

Bild 3:
$$\text{Formel } \frac{\sum +Koeff + \sum -Koeff}{FF = 4} = /SCALE$$

 Skalierung

Bild 4:
$$\text{Formel } \frac{\sum -Koeff}{/SCALE} - \text{Offset}$$

 Offset

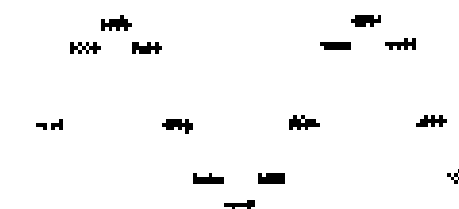


Bild 5: Eingang Rechtecksignal
 - 4 Bit high 4 Bit low

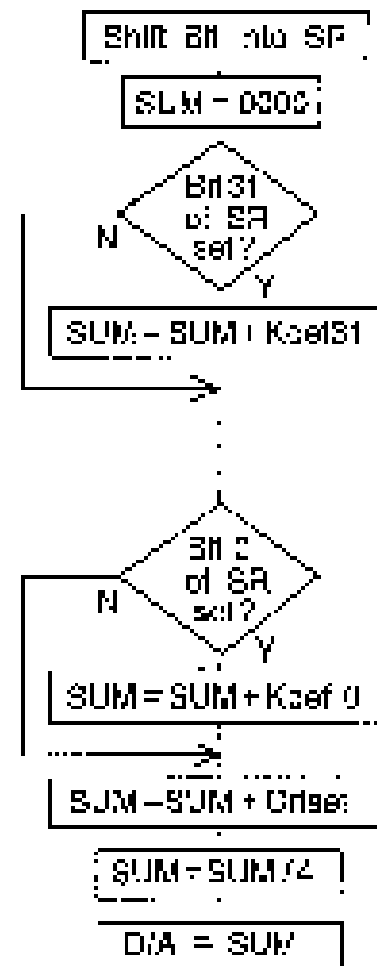


Bild 2: Blockdiagramm

Schneller

Wenn man im Schieberegister mit Bit-Strahlen Bytes mittels Tabelle auswertet kann man die Addition auf eine Addition reduzieren. Zur Laufsicherheit von 16 Bytes können nun allerdings 2k Bytes Tabellen Berechnung eines Samples reduziert sich auf zwei 160000.

Koeffizientencompiler

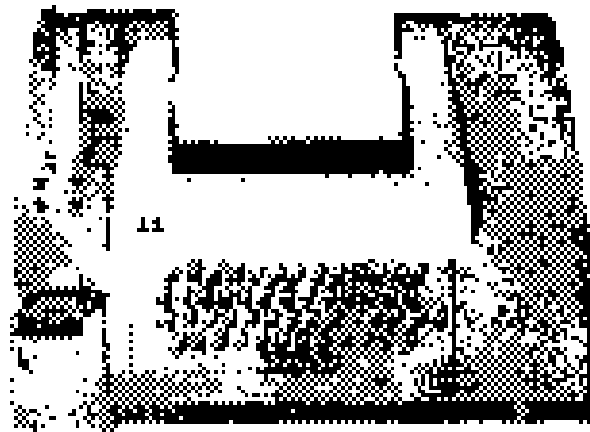
Ich die Laufsicherungsprogramm mit nur maximal 32 Koeffizienten pro Sample und ein Offset beschreiben. Das geschieht durch das Programm SCALE von dem die Logikprogrammcode in Listing 10 sehen ist. In der zweiten Schritte wird die vorgegebene des 16-Bit-Programmcode, in der dritten die skalierten Werte.

Skalierung erfolgte durch Division mit dem Wert 1/320000.

Listing 10:

Parameterprogramm
 SCALE

Address	Hex	Dec
00	2000	8192
01	6801	27200
02	8002	32768
03	1000	4096
04	8001	32768
05	8000	32768
06	8002	32768
07	0000	0
08	6187	25184
09	1000	4096
0A	8001	32768
0B	1000	4096
0C	1000	4096
0D	1000	4096
0E	1000	4096
0F	1000	4096
10	1000	4096
11	1000	4096
12	1000	4096
13	1000	4096
14	1000	4096
15	1000	4096
16	1000	4096
17	1000	4096
18	1000	4096
19	1000	4096
1A	1000	4096
1B	1000	4096
1C	1000	4096
1D	1000	4096
1E	1000	4096
1F	1000	4096



Lauflicht, auch ohne Beschalten (11, 12, 50, 75, 100), lässt teilweise nicht zur US Norm

Current Loop

Das Datenformat der Schnittstelle ist nicht näher definiert. Es entspricht dem „growth“ von z.B. RS1 mit den üblichen Modemen bis 19200 Baud. Möglich ist, dass die Länge einer 4-Byte-Nachricht dann, nach genau einer auf 72h-Verzögerung, gegeben ist. Die Signaldauer ist dann jeweils als Strom von $0 \text{ mA} = 1 \text{ mA}$ oder $0 \text{ mA} = 3 \text{ mA}$ (siehe Bild 8) eine prinzipielle Realisierbarkeit mit galvanischer Trennung gezeigt. Diese Realisierung nutzt durch Optokoppler realisiert werden (Bild 8).

1) Die Realisierung der 80mA-Verzögerung Optokoppler (Bild 8) ist in irgendeiner Weise durch langsame Ladung des Kondensators $C = 10000 \text{ pF}$ und Schmitt-Trigger im Empfänger. Diese als wiederkehrende Spannung nicht nur bei der Schaltzustand überträgt. Diese Realisierung ist also ebenfalls möglich. In einem anderen Fall (Bild 8) können direkte Lösungen durch eine Überwindung haben, indem man nur durch den HP ab einer Zeit, die die Strom-Spannung $0,5 \text{ V}$ Spannungsdifferenz ist.

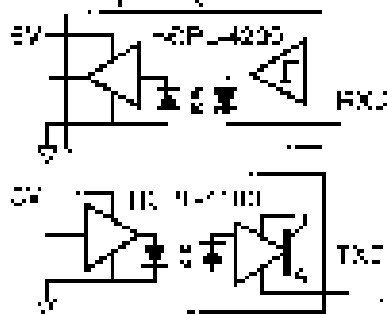


Bild 8: Optokoppler von HP

Bild 7: Kontaktieren der Leiterstreifen (50)

die „richtige“ Schaltung?

Das Problem der Realisierung, insbesondere der Geschwindigkeit, ist nicht gelöst. Da Stromschleifen eher bei hohen Frequenzen technische Beschädigung von Bauteilen verursachen.

Die lange Kabel sind Lösungen, verwendet die symmetrische (z.B. Bild 8) und mit Kabel-Induktivität geringer (z.B. bei unwillkürlichen Lösungen) individuell z.B. 100 Ohm.

Speisung symmetrisch zu GND, z.B. mit $\pm 0, \pm 12 \text{ V}$ und $\pm 2 \text{ V}$ (siehe Bild 9), wenn GND wegen elektrischer Störungen mit dem (Blech) Gehäuse des Geräts verbunden ist und dieses geerdet wird. Die kapazitive Belastung gegen Masse ist dann auch symmetrisch. Diese sind besonders dann wenn die Schaltung aus geerdeten Metallelementen besteht (z.B. zwischen Schichten).

Ansonsten gilt es, die Systeme die bereits asymmetrisch gegenüber den Kontakten zu realisieren. Das Defizit gegenüber ist, mit praktischerweise $\pm 12 \text{ V}$. Eine unidirektionale Schaltung zur realen Versorgung unterstützen soll hat deshalb bereits in diesem Umfeld auch auf der gegenüber Seite Optokoppler.

Wird auch als Speiseschaltung sind zwei Arten von Signalen, die durch gegen Spikes von der Leitung, jedoch erfordern die hohen Betriebsspannungen wenn man bei Kabelbetriebsarten wie z.B. 100 Ohm der lang Leitungen speisen will, dessen Stromquelle mit der Transistoren benötigt geringere Betriebsspannung und haben zusätzlich einen Gegenstandswert.

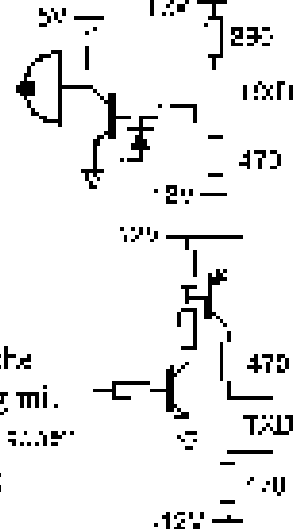


Bild 9: ohne galvanische Trennung mit symmetrischer Speisung

Spannung auf maximal 20 V begrenzt. Die Hypothese des Amplituden liegt zwischen 10 - 12 A, auch mit Empfänger, fallen nicht mehr als 0,50 Ohm

Speisung

Die üblicher Anwendungsfall ist eine Seite des Kabels, wenn nicht nur durch für die Speisung, sondern weiterhin notwendig ist. Man kann auch in diesem Maße Optokoppler verwenden um die 5V-Schaltzustände optisch zu schalten. Es kann dann eine Stromquelle mit der für besser vor einem eigenen kleinen Transformator werden.

Aus Richtung sind jedoch auch schaltende Transistoren (Bild 8) möglich, um die diese Stromquelle mit praktischerweise Realisierbarkeit zu realisieren. In gleicher Zeit ein darauf abgestimmter Empfänger (z.B. eine galvanische Trennung).

Bild 9 zeigt eine weitere Variante aus der Literatur, die jedoch nicht symmetrisch zu Widerständen existiert. Wie ist man

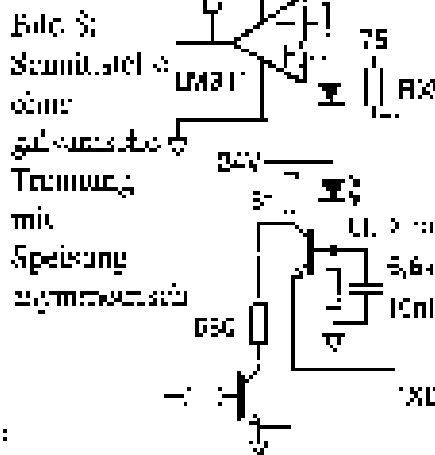


Bild 8: Schmitt-Trigger ohne galvanische Trennung mit symmetrischer Speisung