

embedded

Rafael Deliano
Steinbergstr.37
82110 Germering
Tel 089/8418317

j_r_d@t-online.de
V1.0 (pdf) : 31. Juli 01
V1.1 (pdf) : 29. April 04
V1.2 (pdf) : 14. Jan 07

6

Voransichts- Version

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

- 1 Inhalt
- 2 Manchester
- 4 Prellg
- 5 Mille
- Laufzeitmessung
- 6 Datenkompression
mit LZW
- 10 OCR-Schriften
- 11 E13B Lesegerät
- 14 Barker-Code
- 15 Mehr über LFSR
- 17 Integratoren &
Differenzierer
- 19 Nachträge

AD . ME

Dies ist die erste Ausgabe die
mehr auf Papier erscheint. Die
ung war unvermeidlich, da
kostengünstig ein größerer
on Lesern erreichbar wird.
Der nächste Teil zu fehler-
korrigierenden Codes wurde nicht
rechtzeitig fertig. Aber die Beiträge
zu Schieberegisterschaltungen sind
eine gute Ergänzung des Themas.
Auch der Beitrag über Manchester
sollte als Ergänzung zu FEC gesehen
werden, da er zu den typischen
Anwendungen paßt.

Die Darstellung über Daten-
kompression mit LZW soll im näch-
sten Heft durch eine GIF-Implemen-
tierung vervollständigt werden.

Die Listings sind in
nanoFORTH geschrieben. Für die
Konvertierung in andereFORTH-
Varianten sollte man im nanoFORTH-
Manual GP32 nachlesen.

Datenübertragung mit Manchester-Code

Serielle Übertragung auf gestörten Kanälen erfolgt oft synchron in Paketen mit einem Leitungscode der Taktrückgewinnung ermöglicht.

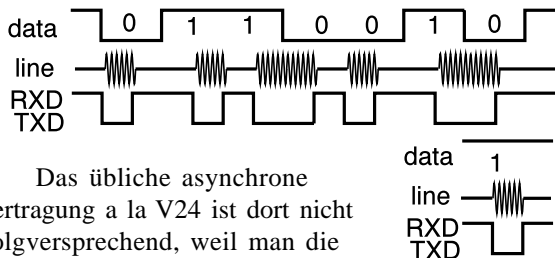


Bild 1: Manchesterpaket

Das übliche asynchrone Übertragung a la V24 ist dort nicht erfolgversprechend, weil man die fallende Flanke des Startbits sauber erkennen muß, damit man das 8. Bit sicher abtasten kann. Zudem ist sie nicht sehr effizient: für 8 Datenbit müssen durch Start- und Stopbit effektiv 10 Bit gesendet werden.

Auf Kabeln werden häufig ternäre Linecodes verwendet die selbsttaktend und gleichstromfrei sind. Binäre Codes haben letztere Eigenschaft zwangsläufig nicht, genügen aber auf anderen Strecken, wie z.B. Funk. Einfach und sehr bekannt ist dafür der Manchestercode, der auch als Biphasen bezeichnet wird. Oft findet man ihn auch unter dem Namen PM, „phase modulation“. Wie man in Bild 2 sieht wird ein 1 Bit durch steigende Flanke in der Mitte des Bits und ein 0 Bit durch fallende Flanke dargestellt. Die Erzeugung erfolgt durch Verknüpfung von Takt und Daten über ein XOR-Gate.

Eine Invertierung von Takt, Daten oder XOR führt allerdings dazu, daß auch die Zuordnung von Flanke zu Bit gedreht ist. Diese invertierte Signalform findet sich in Bild 1 unter Manchester II dargestellt. Manchester II (auch Biphasen-L genannt) wird in dem Feldbus MIL-STD-1553A in Flugzeugen eingesetzt. Um die dort verwendeten Übertrager gleichstromfrei ansteuern zu können, braucht man ein gegenphasiges Signal. Manchester eignet sich dafür im Gegensatz zu den Daten im Rohformat, also NRZ, denn das Energie-

ein enges Frequenzband begrenzt (Bild 3). Ferner hat jedes Bit gleichviel 1 und 0-Anteil, es baut sich also auch bei ungünstigen Daten nicht schleichend DC-Anteil auf.

Andere Codes

Während es für Manchester somit ohnehin schon zwei legitime Versionen gibt, herrscht eine babylonische Sprachverwirrung bezüglich dessen was man als biphasen oder auch biphasen bezeichnet. Das wird dann auch mit Manchester in Verbindung gebracht.

In Bild 4 sind noch einige Varianten dargestellt. Übertragen wird auch hier mit Tastverhältnis 2:1 so daß die Signale auf der Leitung ähnlich aussehen.

Bild 2: Manchester Varianten

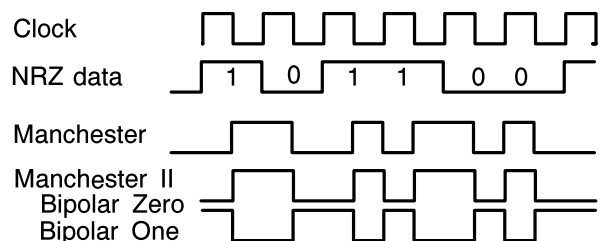


Bild 3: kein Manchester

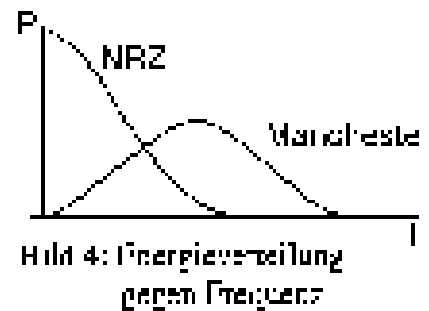
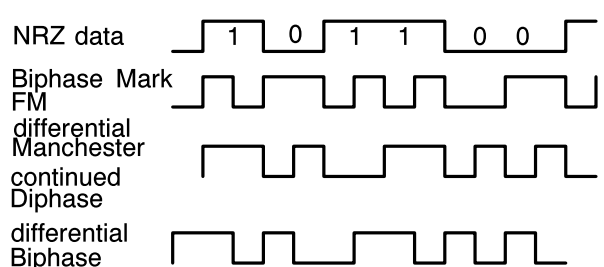


Bild 4: Energieverteilung gegen Frequenz

Controller
Auf kleinen Controller bei Datenraten von 19200 Baud, wie sie bei manchen Funkverbindungen genutzt werden, sind die Übertragung mit Manchestercode im Vergleich zu NRZ durch den Taktanteil um ca. 20% langsamer. In Bild 5 ist die Übertragung von 8 Bit mit geringem Tastverhältnis zu sehen. Die Übertragung wird als biphasen bezeichnet. Hier sei eine bessere Lösung von 8 Bytes angegeben, man kann durch eine Clock im letzten Byte geschickt machen, um den Start des Pakets hier nur anhand der Frequenzänderung am Anfang erkennen. Dadurch ergibt sich die Anforderung, daß das erste Byte ein gesetztes Wort geben muß, hier 00110101, um einen Takt mehr dem Empfänger, sind aber nur 7 Bit effektiv nutzbar. Wenn die Übertragung MSB-First erfolgt, ist angenommen, daß das 2. Bit von 00110101 weniger ist. Neben dem Auftreten des Pakets muß man beim Empfänger auch die Variationen im Takt berücksichtigen, das in Bild 2 deshalb auch dargestellt ist. Falls man nur einen Datenpaket konstanter Länge wie

Bild 2: In 74HC16 sind Schmitt-Trigger, die durch den externen Flankensteilgrad bestimmt, in einer Hysteresis-Schleife den einen Edge des Signals zu erfassen, ob die Signal nicht gleich wieder zurückfällt, wie es in der Spalte von Bild 1 nach einem Reset die Ladegerichtleitung vorgenommen.

Semaphore

Das gesamte System faltet zu einem einzigen Übertragungsweg und

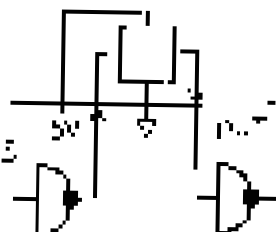
zum nächsten. Es wird hier noch ein ähnliches Timing eines Exports von Daten programmiert, jedoch in der Hauptsache des Programms, was durch den Laufzeit-Übertrag im Testmodus oder ein neues Paket angenommen ist. In diesem Fall wird das Flag gesetzt und es erfolgt Weiterverarbeitung, z.B. Übertragung einer CRC-Prüfsumme. Diese Bearbeitung muß in der Folge zwischen zwei Exports nach erfolgen. Wenn per se kein neue Daten den Speicher

gegenüber schreiben würden

Prellgenerator

Für den reproduzierbaren Test von Software die Signale verarbeitet die bei Flankenwechsel viel zu langsam ist, ist es mit einer Schaltung möglich.

Bild 1 zeigt die Schaltung der Testschaltung



Wie in Bild 1 zeigt, muß man sich auf zwei Bereichen des Zielsystems der Signale, umgeben können um ein Schaltung einseitige Signal erfolgt das, um die für die anlangung des Controllers. Öffnung ist dabei eine 2-pol. Stiftleiste die in Verbinde mit einem Jumper geschlossen wird. Die Schaltung in Bild 2 funktioniert von 5V - 5,5V mit der gleichen Stromaufnahme, kann also aus dem Zielgerät entnommen werden.

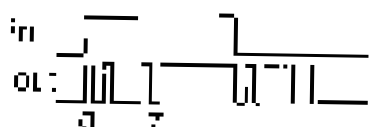


Bild 2: Timing

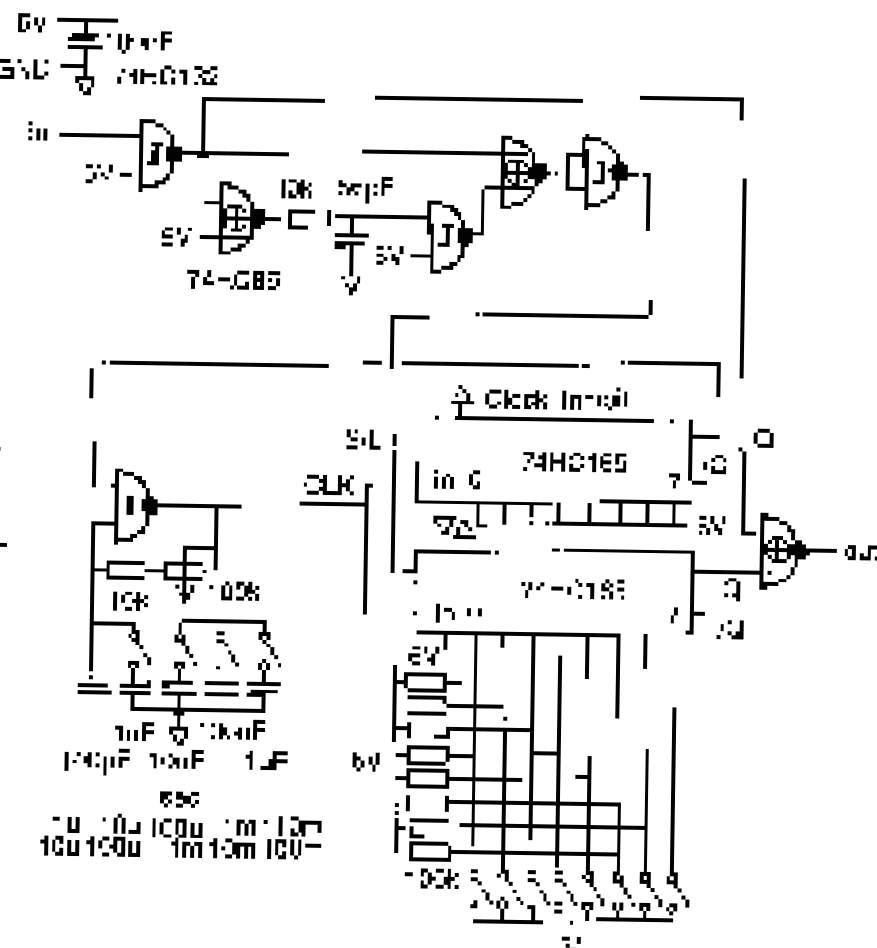


Bild 3 zeigt die Funktion. Schicht bei fallender, als auch bei steigender Flanke werden 8 Datenbits aus einem Schichtregister erzeugt. Die Schaltung ist Pallast also an dem 4-pol. DIP-Schalter programmiert. Die Daten der Test sind über einen D/A-Generator in weiteren Bereichen eines der Test-Zusatzgeräte

Konkretes über einen 4-pol. DIP-Schalter wird die Großzahl der Datenbits des Per erfolgt die Feinjustierung für Schichtregister die sind durch von 1 bis 10 Millisekunden System für Controller sind nach dem Stellen abgehandelt um einen zu einem Zufall des zum 2. Bereich beginnt und 4-pol.

LZW Datenkompression

Dieses Verfahren ist auch für Controller geeignet, denn es wird in V42bis für Modems verwendet. Typisch ist die Verwendung in Grafikdatenformaten wie GIF und TIFF.

Die Vorläufer LZ77 [1] und besonders LZ78 [2] der Israelis Jacob Ziv und Abraham Lempel stellen die Grundlagen dar, waren aber selbst praktisch nicht einsetzbar. Die Weiterentwicklung LZW [3] des Engländers Terry Welch hat sich dafür um so mehr verbreitet. Ein gewisses Problem bei LZW ist das US-Patent von Unisys / Sperry auf das Verfahren [4]. Das hat zeitweise zu Aktivitäten geführt aus LZ77 bzw. LZ78 weitere brauchbare Varianten abzuleiten um das Patent zu umgehen. Dem war jedoch bisher kein nennenswerter Erfolg beschieden.

Prinzip

LZW gehört zur Gruppe „Dictionary based String Compression“. Der Unterschied besteht hier zur einfachen „Character Compression“ bei der immer nur einzelne Symbole, meist Bytes, entsprechend ihrer Häufigkeit in

werden. Codiert man Strings, also Ketten von Symbolen, kann man zwar potentiell höhere Effizienz erreichen, da die Länge der Datenworte nun variabel ist, hat man mit höherer Komplexität zu kämpfen.

Das Verfahren hat mehrere angenehme Eigenschaften. Erstens wird kein Vorwissen über die Häufigkeitsverteilung der Eingangsdaten benötigt, also etwa eine fixe Häufigkeitstabelle. Die Vorgaben aus solchen Tabellen passen meist nicht zu den kurzen Datensätzen die man verarbeiten will. Der Algorithmus baut sich das Codebuch deshalb selbst auf.

Zweitens erfolgt die Bearbeitung in einem Durchlauf, bei dem ein sequentieller Datenstrom eingelesen und ausgegeben wird. Andere Verfahren müssen den Datensatz zweimal lesen und senden erst beim zweiten Durchlauf.

Drittens muß die Codierungstabelle nicht separat übertragen werden, sondern ist implizit im Ausgangsdatenstrom enthalten.

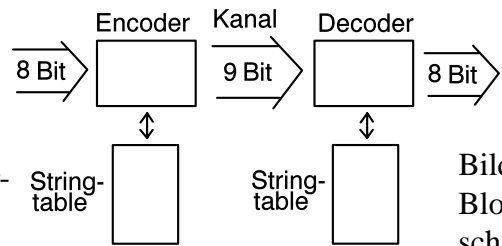


Bild 1: Blockschaltbild

Der Eingangssymbole werden meist Bytes verwendet. Die Wortbreite der Ausgangstoken ist etwas größer als die der Eingangssymbole. Abhängig von Implementierung meist 9 - 14 Bit. Eine typische Obergrenze ist 12 Bit.

Die erreichbare Effizienz hängt natürlich von der Struktur der Daten ab. Zufallszahlen sind praktisch nicht komprimierbar. Tabelle 1 zeigt praktische Beispiele für V42bis. In weniger realen Fällen, bei völlig redundanten Daten wurden auch schon 10% erreicht.

Tabelle 1: V42bis

Umfang Daten nach Codierung:

100%	Zufallszahlen
65%	Objektcode
60%	gemischte Daten
50%	Text
30%	Grafik
30%	Sourcecode

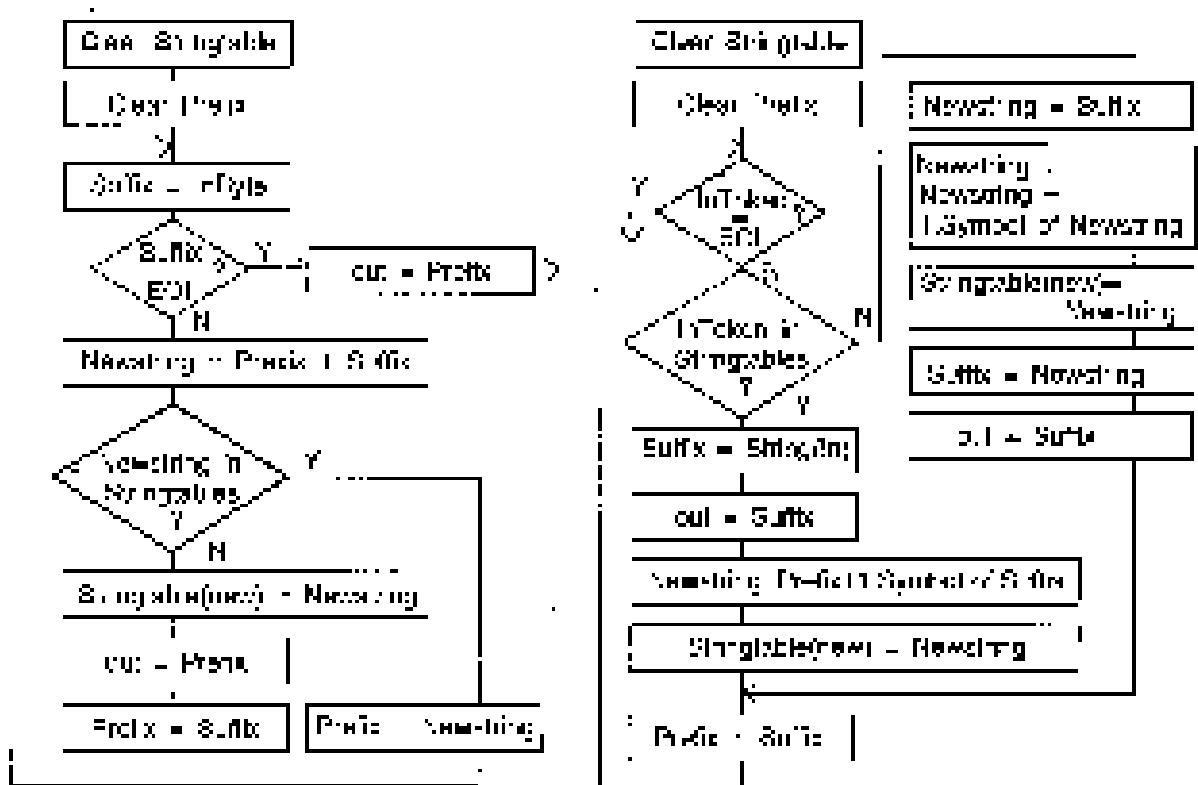
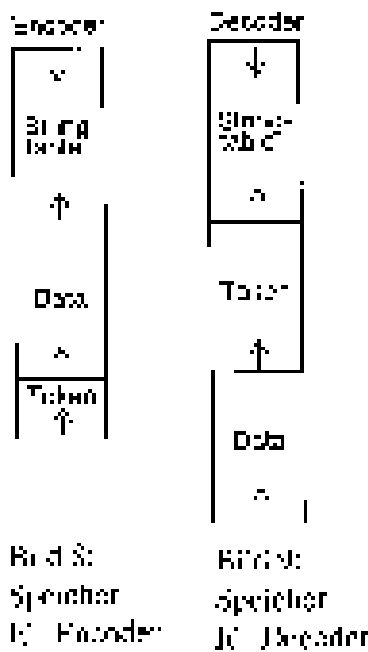


Tabelle 2: Anschluss von Textarten

id	prefix	bezeichnung	abzug in tablen	abzug in	bezeichnung
0000		G			
0001	G	CH	0100 CH	0000	
0002	H	HA	0100 HA	0000	
0003	I	CA			
0004	I	CC	0110 CC	0100	
0005	I	CD	0110 CD	0000	
0006	I	CE			
0007	I	CF			
0008	I	CG	0110 CG	0100	
0009	I	CH			
0010	I	CI			
0011	I	CJ			
0012	I	CK			
0013	I	CL			
0014	I	CM			
0015	I	CN			
0016	I	CO			
0017	I	CP			
0018	I	CQ			
0019	I	CR			
0020	I	CS			
0021	I	CT			
0022	I	CU			
0023	I	CV			
0024	I	CW			
0025	I	CX			
0026	I	CY			
0027	I	CZ			
0028	I	CA			
0029	I	CB			
0030	I	CC			
0031	I	CD			
0032	I	CE			
0033	I	CF			
0034	I	CG			
0035	I	CH			
0036	I	CI			
0037	I	CJ			
0038	I	CK			
0039	I	CL			
0040	I	CM			
0041	I	CN			
0042	I	CO			
0043	I	CP			
0044	I	CQ			
0045	I	CR			
0046	I	CS			
0047	I	CT			
0048	I	CU			
0049	I	CV			
0050	I	CW			
0051	I	CX			
0052	I	CY			
0053	I	CZ			
0054	I	CA			
0055	I	CB			
0056	I	CC			
0057	I	CD			
0058	I	CE			
0059	I	CF			
0060	I	CG			
0061	I	CH			
0062	I	CI			
0063	I	CJ			
0064	I	CK			
0065	I	CL			
0066	I	CM			
0067	I	CN			
0068	I	CO			
0069	I	CP			
0070	I	CQ			
0071	I	CR			
0072	I	CS			
0073	I	CT			
0074	I	CU			
0075	I	CV			
0076	I	CW			
0077	I	CX			
0078	I	CY			
0079	I	CZ			
0080	I	CA			
0081	I	CB			
0082	I	CC			
0083	I	CD			
0084	I	CE			
0085	I	CF			
0086	I	CG			
0087	I	CH			
0088	I	CI			
0089	I	CJ			
0090	I	CK			
0091	I	CL			
0092	I	CM			
0093	I	CN			
0094	I	CO			
0095	I	CP			
0096	I	CQ			
0097	I	CR			
0098	I	CS			
0099	I	CT			
0100	I	CU			
0101	I	CV			
0102	I	CW			
0103	I	CX			
0104	I	CY			
0105	I	CZ			
0106	I	CA			
0107	I	CB			
0108	I	CC			
0109	I	CD			
0110	I	CE			
0111	I	CF			
0112	I	CG			
0113	I	CH			
0114	I	CI			
0115	I	CJ			
0116	I	CK			
0117	I	CL			
0118	I	CM			
0119	I	CN			
0120	I	CO			
0121	I	CP			
0122	I	CQ			
0123	I	CR			
0124	I	CS			
0125	I	CT			
0126	I	CU			
0127	I	CV			
0128	I	CW			
0129	I	CX			
0130	I	CY			
0131	I	CZ			
0132	I	CA			
0133	I	CB			
0134	I	CC			
0135	I	CD			
0136	I	CE			
0137	I	CF			
0138	I	CG			
0139	I	CH			
0140	I	CI			
0141	I	CJ			
0142	I	CK			
0143	I	CL			
0144	I	CM			
0145	I	CN			
0146	I	CO			
0147	I	CP			
0148	I	CQ			
0149	I	CR			
0150	I	CS			
0151	I	CT			
0152	I	CU			
0153	I	CV			
0154	I	CW			
0155	I	CX			
0156	I	CY			
0157	I	CZ			
0158	I	CA			
0159	I	CB			
0160	I	CC			
0161	I	CD			
0162	I	CE			
0163	I	CF			
0164	I	CG			
0165	I	CH			
0166	I	CI			
0167	I	CJ			
0168	I	CK			
0169	I	CL			
0170	I	CM			
0171	I	CN			
0172	I	CO			
0173	I	CP			
0174	I	CQ			
0175	I	CR			
0176	I	CS			
0177	I	CT			
0178	I	CU			
0179	I	CV			
0180	I	CW			
0181	I	CX			
0182	I	CY			
0183	I	CZ			
0184	I	CA			
0185	I	CB			
0186	I	CC			
0187	I	CD			
0188	I	CE			
0189	I	CF			
0190	I	CG			
0191	I	CH			
0192	I	CI			
0193	I	CJ			
0194	I	CK			
0195	I	CL			
0196	I	CM			
0197	I	CN			
0198	I	CO			
0199	I	CP			



Tabelle

Frage: als Implementierung
 mal mit Strings (1000) und mit Tokens
 (1000) mal länger 250 Bytes arbeiten
 können.

Frage: woher die Variable
 wie in Bild 7? bei der werden die Werte
 einer Tabelle mit Werten von 1-4
 vergeben. Wenn jeder die ersten 10 die
 die Adresse der Strings in die zweiten
 8 Bit seine Länge. Die Bytes folgen
 sehr schnell was man nach oben
 Variable (1000) und die Länge (1000)

Frage: Token Prefix, Suffix und
 Newline müssen ebenfalls sehr lang
 Strings enthalten können. Diese
 Speicherung in Variablen erfordert
 Aufmerksamkeit was man zeigen. In
 den Daten Speicher der Performance
 (1000) ist wichtig. Die geeignete Anordnung
 in den Decoder ist das man die
 Strings nicht selbst in die
 Tabelle übernehmen kann.

Beachtung: (1000)
 wie in Bild 7 (1000) ist
 implementiert wie abgebildet.

In Decoder wird die gleiche
 Anordnung von Prefix und Suffix
 verwendet.

Puffer

Neben dem Speicher für die
 Tabelle werden auch Puffer für
 Token und Daten benötigt. Bild 8 zeigt
 eine geeignete Anordnung für den
 Encoder. Bild 9 für den Decoder.

Grund der Implementierung
 sind man zwei Token Puffer
 für Token (1000) und Daten (1000) und
 uniform Token, bei denen alle Bytes
 den gleichen Wert haben. Das Ergebnis
 für den Speicherbedarf der Tabelle
 zeigt Bild 11: bei Zufallszahlen muß die
 Speichergröße mehr als 6 mal so
 groß wie im Beispiel von Bild
 10 sein. In der Tabelle von Bild
 10 sind Token und die Token
 25% länger als die Signale
 von 100. Kompression würde gar
 keinen Sinn machen.

Bei Zufallszahlen ist der
 Encoder nicht so sehr langsam. Bei der
 gewählten einfachen Implementierung
 brauchen 1000 Bytes Zufallszahlen
 30000 Bytes in einem 2MBit Speicher
 (1000) 30% uniformen Daten. Möglich
 auch die Encoder nur ein Token, der
 Decoder arbeitet schneller.

Diese speziellen Daten sind
 möglich durch einen Decoder
 zentral des ungenutzten Bereiches der
 Speicher (1000) mit Tabelle fertig
 mit dynamisch zugeordnet. In die
 Speicher (1000) im ungenutzten Teil
 gefügter Kompression von Zufalls-
 zahlen nicht viel schneller gewesen
 der Decoder (1000) können sich
 nicht so sich stark überlegen.

In Decoder (Bild 9) sind
 3 Token ungenutzte. Die Länge des
 erplanten Token (1000) bestimmt
 den Umfang der beiden Token
 Puffer.

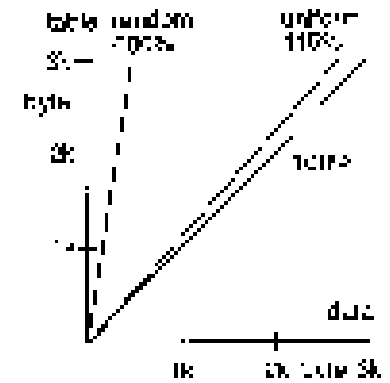


Bild 11:
 Speicherverbrauch Token

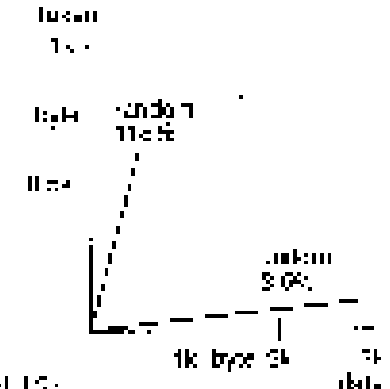
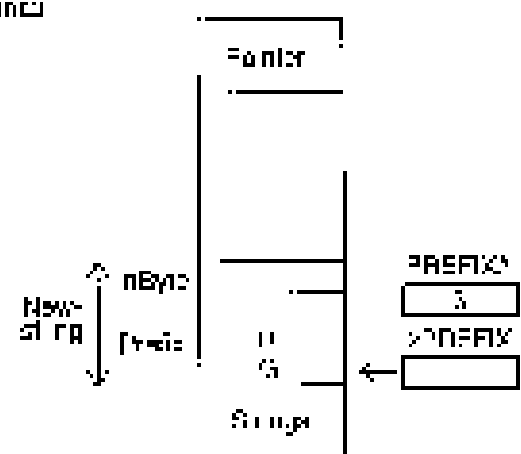


Bild 12:
 Speicherverbrauch Token

- [1] Ziv, J., Lempel, A. Universal algorithm for lossless data compression. 1977 IEEE Transactions. JU 25 No. 3
- [2] Ziv, J., Lempel, A. Compression of individual sequences via context-sensitive algorithm. Universal algorithm for sequences data compression. 1977 IEEE Transactions. JU 25 No. 3
- [3] Welch, J. A. Technique for high performance data compression. 1984 IEEE Computer Vol. 17, no. 5 June 1984
- [4] 953 US-Patentnummer 4,556,000

Bild 10: Stringregister



OCR-Schriften

Bei codes werden nicht Existenz dem Schließen der OCR-Entwicklung in den 50er Jahren. Wenn man sich ein Barcode-Label anschaut (Bild 1), sieht man fast, daß fast immer unter dem Barcode für die Maschine auch die Ziffern für den Menschen gedruckt werden. Unglücklich war es nur, daß diese Ziffern nicht durch Automaten lesen lassen. Man erkannte aber schnell, daß Zeichensätze mit konventioneller Typographie mit den damaligen technischen Mitteln nicht ausgewertet werden konnten.



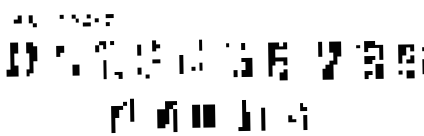
Bild 1: Barcode

Aus Anlaß der immer mehr die Zeichen angestrichelten Maschinen auswertete, konnte man sich vorstellen, daß man vollautomatische OCR-Auswertung durch die Maschinen ermöglichen könnte. Die Lösung lag in der Entwicklung von OCR-Schriften, die für die Maschinen lesbar sind, aber auch für den Menschen verständlich.

Magnetchriften

In den 50er Jahren fanden sich keine hochentwickelten optischen Sensoren zur Verfügung. Es begann also mit der Entwicklung der Magnetkassen ("Magnetic Ink Character Recognition" MICR). Man bedient sich mit Tintenstrahlmaschinen zur Zeichnung magnetischer Tinten. Dadurch sind sie von bestimmten Tintenstrahlmaschinen ("Dot-Matrix-Druckmaschinen") zu unterscheiden, die eine geringe Schärfe und Kratzenempfindlichkeit in Anwesenheit von Scheuertintenstrahlen nicht als ungewollt.

Bild 2: MICR-Schriften



0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19

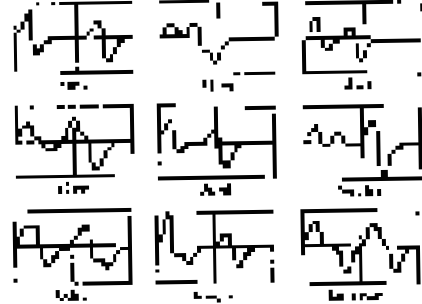


Bild 3: Signalform MICR

Die erste bekannte magnetische Code war MICR (Bild 3), der von Stanford Research Institute und Lockheed für die American Bankers Association (ABA) entwickelt wurde. Er wurde immer noch in den USA verwendet, allerdings heute nicht optisch, sondern durch magnetische Lesegeräte. Am ursprünglichen Lesegerät angepaßt, sind auch heute noch MICR dargestellt. Die Signalform wurde hier durch den Magnetismus, abgelesen. Die Codierung der Ziffern konnte durch die einfache, durch gewöhnliche Tintenstrahlmaschinen als magnetische Schicht aufgetragen, mit einer sehr geringen Auflösung, die keine hohen Anforderungen an die Lesegeräte stellt.

Man entwickelte deshalb eine Form der magnetischen digitalen Codierung. Der Code von MICR (Bild 4) verwendet eine Ziffer pro Ansatz mit einem Zeichen direkt in einem 5x500 Zeichen Code, was nicht möglich war. Der Vorteil besteht darin, daß MICR, das nur durch Lesegeräte der Buchstabenlänge möglich ist, jetzt

Bild 4: MICR-Bicode

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15



Bild 5: Signalform MICR-B

per magnetischen Lesegeräte des Textes immer. Ein Zeichencharakter funktioniert aus, wenn man sehr genau auswertet. Es war aber durch keine Verkleinerung gefordert.

Anders aber wurde die Entwicklung von MICR-B (Bild 5) durch den IBM 3600-Drucker im Jahr 1964. Es wurde als Barcode-Code entwickelt, der besteht aus 9 Zeichen einer Abfolge der Codierung, die die Abfolge der Zeichen weiterführt mit einem Magnetkopf. Die Abfolge der Zeichen wird durch die Abfolge der Zeichen weitergeführt. MICR-B war in Europa weit verbreitet.

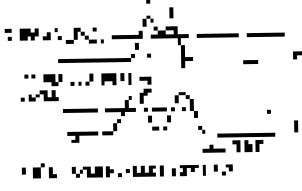
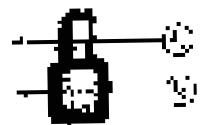
Die Zeichen-Systeme sind für die Abfolge der Zeichen weitergeführt. Die Zeichen-Systeme sind für die Abfolge der Zeichen weitergeführt.

OCR

Alle Magnetkassen sind auch optisch auswertbar. Da das Lesegeräte sind, werden die Zeichencharaktere durch die Zeichencharaktere weitergeführt.

Alle Magnetkassen sind auch optisch auswertbar. Da das Lesegeräte sind, werden die Zeichencharaktere durch die Zeichencharaktere weitergeführt.

OCR-A (Bild 6) wurde 1966 von ANSI entwickelt und ist eine 12 Zeichen lange Code. Auf die Zeichencharaktere immer noch zu finden. OCR-B (Bild 7) entstand 1968 als bessere Typographie für die Frager im Auftrag der European Computer Manufacturers Association (ECMA) 1971 überarbeitet und wurde als OCR-Schrift zum Code.



abcdefghijklmnopqrstuvwxyz
 ABCDEFGHIJKLMNOPQRSTUVWXYZ
 1234567890 <.:;?@#

LASER

Light von Lasergeräten

Bl. 18: Dekor

erforderter und somit ein Bestandteil der Markteinführung.

Techno-Kitsch

Für die OCR-Revolution wurde von Anfang an sich zwei Hauptaufgaben gestellt: die sie als Spielzeug zu vermarkten und ein Publikum zu identifizieren zu suggerieren. Bild 11 zeigt, warum die Zeichenbreite oft willkürlich erweitert und verkleinert wird, so daß immer die automatische Auswertemaschine schon zu gekommen ist. In diese Firma schenken den Hürden der Computerkriterien ein Sachleben beschließen zu sein.

- [1] Fischer „Optical Character Recognition“ Special Books 1969
- [2] Steinbuch „Handbuch der Nachrichtentechnik“ Springer 1967
- [3] Frank „Kryptomatische Verfahren“ Fischer 1967

Bl. 6: OCR-A
 ABCDEFGHIJKLMNOPQRSTUVWXYZ
 1234567890 <.:;?@#

Bl. 7: OCR-B
 abcdefghijklmnopqrstuvwxyz
 ABCDEFGHIJKLMNOPQRSTUVWXYZ
 1234567890 <.:;?@#

Bild 4: Intralesches Lesegerät



Simulation eines E-13B Lesegeräts

Niemand wird heute ein Lesegerät für die E-13B Schrift mit Magnetkopf neu entwickeln. Rechnerimulation eines solchen Geräts ist jedoch nutzlos, weil man mit geringem Aufwand an einem überschaubaren Problem die Funktion eines Mastererkenntungsverfahrens studieren kann.

Bild 1 zeigt ein Steuerprogramm für ein komplettes E-13B-Lesegerät, ein General Electric vom 1960. Die Elektronik, die TTL Transistoren und die Proben, war nicht geeignet für die Schichten unter die Lötlöt über zu führen die Ringkerne mit einer die Verknüpfungsergebnisse in einem Konsole geführt werden.

Symbole

Der Zeichensatz (Bild 2) beschränkt sich auf die Zahlen von 0-9 sowie die 4 Sonderzeichen die in Software mehr als die Buchstaben A-Z codiert werden:

A	trans.	Leertst.
B	manant	Rechn.
C	vers	Kont.
D	rest	Stich

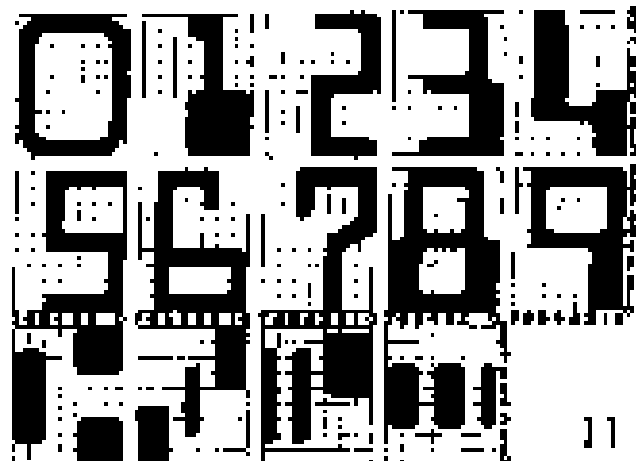
Für Zeichen entsprechen anderen, wie 100 Matrix. Man beachtet, daß die Zeichen von nach unten links gelesen werden sollen. Deshalb stehen sie auch

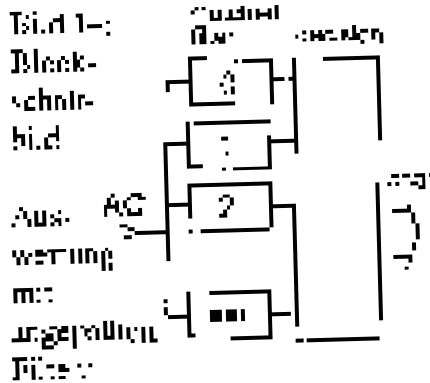
Bild 2:
Zeichensatz E-13B

nachfolgend in der Matrix. Der Bereich ist so ausgeführt, daß nichts ist eine Karte zur nonstandard schwarzen Bildpunkte vorhanden ist, die eine ausgeprägter Eingangsleitung erzeugt die die Detektierung des Aufbaus ermöglicht. Im Symbolen Coding (MERC) wird eine 7x5 Matrix verwendet, in dieser über jeder Bildpunkt mit 2 Bit codiert. Daher wird keine Schwärzung in Graustufen angewandt.

1	1000
2	0000
3	0001
4	0010

Jede Zeile besteht da ist aus 2 Bytes in denen die Zeichen 14. Die belegt sind die Zeichen benötigt aus 16 Bytes Speicher. Für die publikation Darstellung von der Lernprogramm aus dem ROM im RAM kopieren. Das kann man durch die weiteren Auswertung nach Unlöslichkeiten erfolgt, um die Elemente der Erkennung zu prüfen.





Im Blockschaltbild ist die Auswertung mit angelegtem Filter dargestellt. In der Zähler- und Auswertung-Block sind die entsprechenden Funktionen dargestellt.

Syntax

In politischen Auseinandersetzungen werden sehr politische Zeichen verwendet. Die Zeichen sind oft sehr komplex und stellen die Komplexität der Situation dar. In diesem Fall wird man z.B. gewisse Zeichen vor Abschluss der Verhandlung nicht beachten. Man kann sich durch Zeichen in einer Liste, dem Bank- und Zahlensystem, feststellen, ob es sich um ein politisches Zeichen handelt. Die Zeichen sind oft sehr komplex und stellen die Komplexität der Situation dar.

Die Errechnung der Wahrscheinlichkeit der Zeichen liegt nicht nur in der Wahrscheinlichkeit der Zeichen, sondern auch in der Wahrscheinlichkeit der Zeichen. Die Wahrscheinlichkeit der Zeichen ist ein Maß für die Wahrscheinlichkeit der Zeichen. Die Wahrscheinlichkeit der Zeichen ist ein Maß für die Wahrscheinlichkeit der Zeichen.

[1] Final Report of the Joint Committee on the Organization of the Federal Government, 1962

Barker-Codes

Im vorhergehenden Beitrag als Beispiel für einen Code der optimal für matched filter anwendbar ist angesprochen, sollen hier die anderen Varianten kurz vorgestellt werden.

Es gibt nämlich anscheinend nur die in Bild 1 dargestellten. Wobei jeder Code doppelt verwendbar ist, da man die Reihenfolge umdrehen kann.

Für die Darstellung der Korrelation ist hier die übliche „analoge“ Darstellung gewählt die mit binären Daten zu Dreiecken führt. Sie entsteht, wenn

man beide Codes kontinuierlich gegeneinander verschiebt. Wenn bei digitaler Simulation die Verschiebung ruckartig um 1 Bit erfolgt entstehen Treppchen.

Von praktischer Bedeutung sind natürlich die langen Codes, weil hier die Spitze besonders ausgeprägt ist. Die 11-stellige Version wird in beiden

Varianten in der Uko-Schnittstelle von ISDN zur Rahmensynchronisation verwendet.

Wenn man viele verschiedene Codes braucht, z.B. bei Spread Spektrum Systemen verwendet man Gold-Sequenzen. Auch Kasami- und Walsh-Sequenzen werden empfohlen.

[1] Barker „Group Synchronizing of Binary Digital Systems“ in: Jackson „Communication Theory“ Academic Press 1953

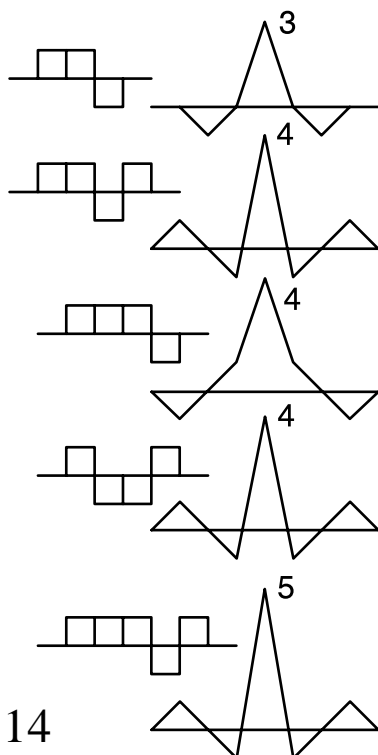
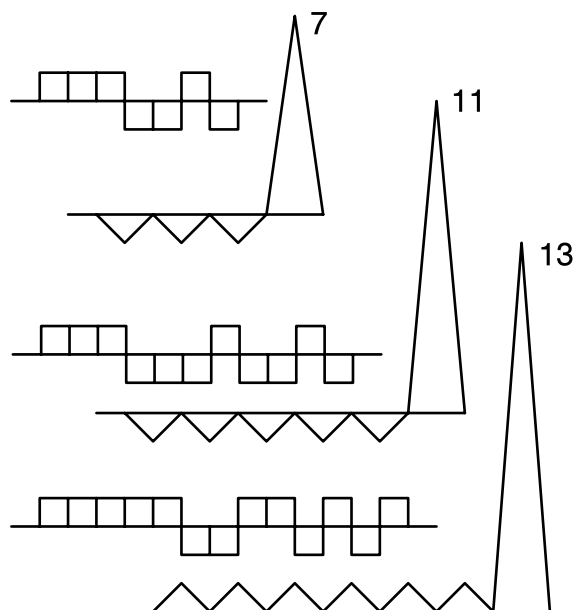
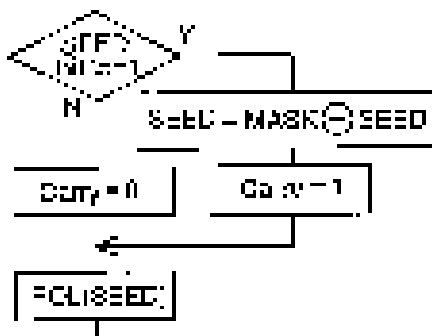


Bild 1: Barkercodes und ihre Korrelation



LFSR-Generatoren in Software

Die Takt für einen 16 Bit Generator Typ I zeigt Bild 1. Jede Taktecke gewöhnlich in Bild 3 in der Tabelle von Schritt 1. Das im Durchlaufprogramm Bild 2 gezeigt, initialisiert den Wert des 16 Bit und initialisiert ihn, wenn eine neue Sequenz gestartet ist. Der Ausgang des Moduls ist 1, wenn die Summe ungerade ist. Das steht man im untersten Bit des Akkus, das kann man mit Shiftregister im Beispiel Schritt 1 und man den als 16 Bit Register steuert ein oder führt.



Jedes Polynom überführt in einem digitalen Programm, aber man kann die Übergang durch eines Parametergenerators automatisieren.

Teil 1, Bild 3: 16 Bit mit gleichem Polynom, aber wie das Durchlaufprogramm in Bild 1 zeigt, ist die Feedback für die 16 Bit. Man kann die Polynom in der Konstante SEED definieren, die die 16 Bit, und dann ein ungerades Programm für 16 Bit Register in jedem konfigurieren.

Bild 4: Flowdiagramm Typ I

Bild 3: Typ I

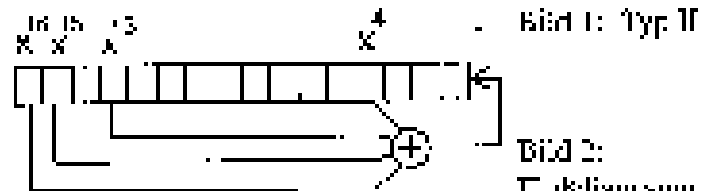
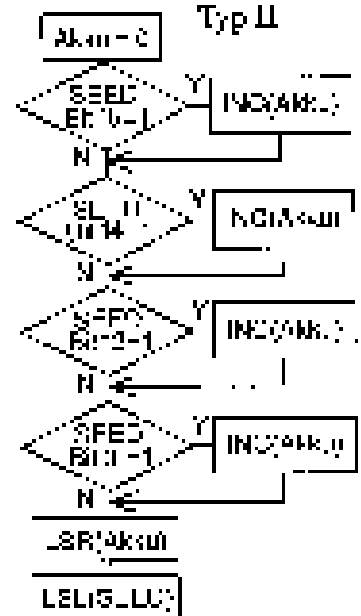


Bild 1: Typ II

Bild 2: Flowdiagramm Typ II



Polynome für m-Sequenzen

Wenn man die Verteilung der üblichen Timings der Tabellen mitteilt, wird es eine gute Variante mit einer Tabelle geben. Man muß aber Polynome für die Sequenzen selbst erzeugen.

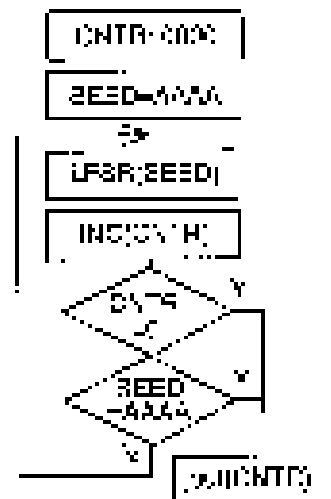
Werte derartige Polynome gibt es überhaupt nicht. Die gibt eine Formel die diese Zahl nicht bestimmt. Sie beruht auf der Inversen Pol-Funktion und ist etwas schwierig zu berechnen. Die Definition der Eigenschaft (1) in Tabelle 1 (vgl. Beitrag in dem 1. Heft 2).

Damit wird man die diese nicht mögliche Zahl von m-Sequenzen für die Zahl verändert. bekannt die gewöhnlich in polynom Polynome überführt berechnet. Die dann die Lösung. Vorgeschrieben nimmt man diese Zahl und prüft sie es ist. Nur bei sehr geringen Wertungen kann man über die Kombinationen abstrahieren und die Testen durchführen. Bei der Fähigkeit. Eine andere Lösung? Man muß

nur gehen und man einen Zahlenzahlengenerator auswählen. Die Schritt zeigt dem, wenn man sich der besten Chance haben.

Die Testmethode Testcase Methode ist stapel (Bild 1). Man hat ein 16 Bit LFSR-Generator (Typ I) beständig mit einem Register mit 4444 und initialisiert man 16 Bit 23er mit 000. Dann wird man Generator und 23er 1000. Erste Anmerkungen ist, wenn der Zähler wieder 000 erreicht ist, weiter. Ähnlich können ist, wenn der Inhalt des LFSR die Muster 4444 wieder erreicht hat. In beiden Fällen gibt man ein oder der Zähler. Die ist ein in Sequenz die Zustände mit 000. Sind immer

Bild 1: Testcase Suche



weil man weiß, daß der Zählerwert bei dieser Anwendung von Ende FFFF ist.

Der Fehlerfall ist möglich, daß bei steigender Registerlänge der Zählerwert mit 2^n steigt. Wenn das ein Problem wird gibt es Kriterien mit denen man die Leistung der Suche etwas steigern kann. Die Zahl der überprüften Tests muß groß sein.

(2, 4, 6, ...). Das halbiert die Zahl der Möglichkeiten. Aus jeder m-Sequenz läßt sich durch Umkehrung der Reihenfolge der Bits („reverse“) eine umgekehrt zählende m-Sequenz direkt ableiten. Damit zählt jeder Treffer doppelt. In Tabelle 2 sind deshalb (außer für n=2, denn 111 = 111) alle Zahlen gerade.
 [1] Wustmann „Erzeugung von Pseudo-Zufallsfolgen mit binären Schieberegistern“
 Elektronik“ 18/1982

Tabelle 1: Zahl der m-Sequenzen abhängig von Wortlänge

n	m-seq	%	n	m-seq	%	n	m-seq	%
2	1	25,0	12	144	3,5	22	120032	2,9
3	2	25,0	13	630	7,7	23	356960	4,3
4	2	12,5	14	756	4,6	24	276480	1,6
5	6	18,8	15	1800	5,5	25	1296000	3,9
6	6	9,4	16	2048	3,1	26	1719900	2,6
7	18	14,0	17	7710	5,9	27	4202496	3,1
8	16	6,3	18	8064	3,1	28	4740632	1,8
9	48	9,4	19	27594	5,3	29	2300976	0,4
10	60	5,9	20	24000	2,3	30	17820000	1,7
11	176	8,6	21	84672	4,0	31	69273666	3,2
						32	67108864	1,6

Integration & Differenzierer

In Heft 2 wurden bereits die simplen, schnellen Varianten mit Anwendung in PID-Reglern beschrieben. Für Datenanalyse könne aber bessere Verfahren nützlich sein. Zumal der Programmieraufwand gering ist, da es sich immer um FIR-Filter handelt.

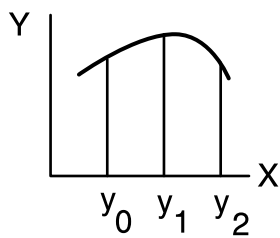


Bild 1: Integrator mit 2 Stützstellen

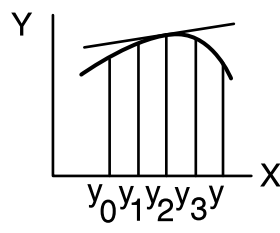


Bild 3: Differenzierer auf y2

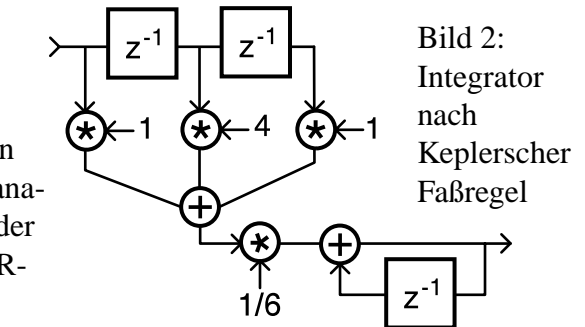


Bild 2: Integrator nach Keplerscher Faßregel

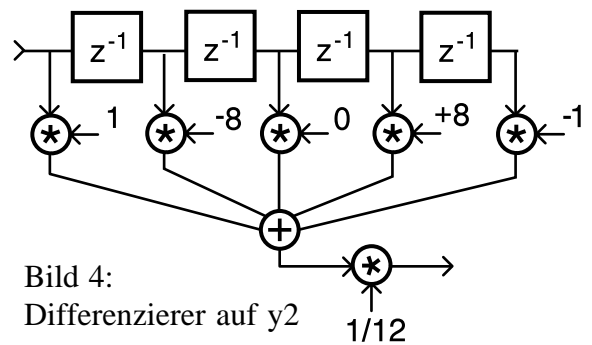


Bild 4: Differenzierer auf y2

Integration:

Hier soll bekanntlich die Fläche unter einer Kurve bestimmt werden. In der Praxis findet sich dafür manchmal auch der Begriff Quader an, wobei es für die Volumenbestimmung von Körpern durch entsprechende Regeln ausreicht. In der Filter-Anwendung sind die Beispiele im gleichem Ansatz wie bei der Rechteck- und der Trapezregel vorzuziehen, da nur 2 Samples für die Trapezregel sind es bereits 3 Samples (Bild 1, Tabelle 1), während sich nur mit 2 Samples besser arbeiten lassen sollte. Die Formel berechnet auch nur die Fläche zwischen y0 und y1. Die einen konventionell arbeitenden Filterpunkt muß man die verbleibende Fläche schrittweise integrieren. Im Falle der Trapezregel folgt ebenfalls ein ähnliches Integrationsverfahren (Bild 3). Den Newton-Cotes

Formeln höherer Ordnung sind etwa 5 Samples sind anzugeben, daß die maximale Genauigkeit nicht dem Rechnerwert entspricht, weil Verfahren wie Runge-Kutta oder Runge-Kutta-Methode mit bereits negativen Coeffizienten konvergieren manchmal nicht mehr, weshalb Newton-Cotes Formeln nach höherer Ordnung nicht verwendet werden.

Differenzierer

Diese Funktion soll die Ableitung an einem Punkt abschätzen (Bild 3). Die Funktionsänderung erfolgt auch hier durch die FIR-Filter (Bild 3). Da die Daten meist von einem Analog-Digital-Wandler kommen, wird die Qualität besser wenn man mehr Samples verwendet. In Tabelle 2 hat man die Auswahl an welchem Punkt die Ableitung berechnet werden soll.

Die Ableitung gibt eine Bestimmung der Steigung an. Bei einer PID-Regel muß man aber die y2 schätzen, weil man zukünftige Samples auch noch kennen will. Auch in der Endposition eines geschlossenen Regelkreises muß man Schätzungen mit den Endwerten durchführen, auch wenn man den Test mit y1 positiv durchrechnet.

Theoretisch kann man gewisse Ableitungen durch unregelmäßige Filter annehmen beschreiben. Praktisch ist es nur weniger Probleme berechnen, aus der Originalform nur rechnerische Polynome. Die zweite Tabelle 3 oder dritte Abbildung, Tabelle 4, kann berechnen. Auch hier gibt es bessere Verfahren wie Splines oder Runge-Kutta. Differenzierer messen aber gleichmäßig über zwei Ergebnisse, um nicht falsche Werte zu liefern. Die Ableitung ist schätzungsweise nicht möglich.

- 1) Siegel, Applied numerical Methods, 2nd Edition, Macmillan, 1964
 2) Engels, Muller, Reiter, Numerik-Algorithmen, 4th Edition, 1995

Tabelle 1: Koeffizienten für Integration nach Newton-Cotes

	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
1/2	1	1							
1/3	1	4	1						
3/8	1		3	1					
8/25	7	24	14	24	7				
1/20	10	75	70	70	75	10			
1/147	43	216	27	272		249	41		
1/1280	751	3072	144	2052	2880	1512	3840	751	
1/10752	983	5888	-870	7128	4340	10516	-471	5081	189

Computer
 Koeffizienten-Paare
 (Koeffizient, Schrittweite)

Tabelle 2: Koeffizienten für Ableitung

	y_0	y_1	y_2
y_0'	1/2h	-1	1
y_1'	1/2h	1	-1
y_2'	1/6h	-3	3
y_3'	1/6h	3	-3

Tabelle 4: Koeffizienten für 3. Ableitung

	y_0	y_1	y_2	y_3	y_4	y_5
y_0'''	1/6h^3	-3	3	-1	1	
y_1'''	1/6h^3	3	-3	1	-1	
y_2'''	1/24h^3	-9	27	-27	9	
y_3'''	1/24h^3	9	-27	27	-9	
y_4'''	1/60h^3	-15	45	-45	15	
y_5'''	1/60h^3	15	-45	45	-15	

	y_0	y_1	y_2	y_3	y_4
y_0''''	1/24h^4	-6	12	-6	
y_1''''	1/24h^4	6	-12	6	
y_2''''	1/120h^4	-30	90	-90	30
y_3''''	1/120h^4	30	-90	90	-30
y_4''''	1/720h^4	-42	126	-126	42
y_5''''	1/720h^4	42	-126	126	-42

Tabelle 3: Koeffizienten für 2. Ableitung

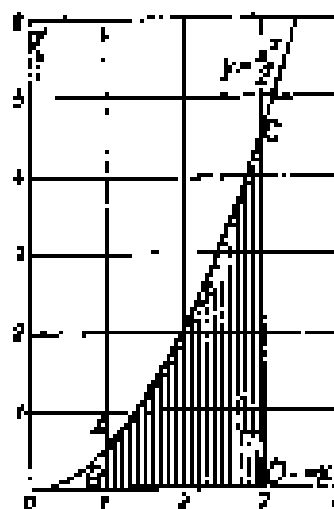
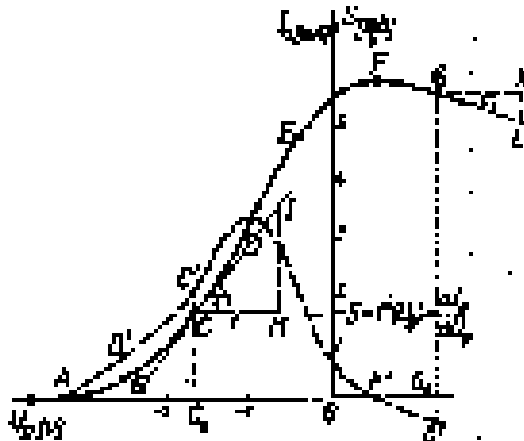
	y_0	y_1	y_2
y_0''	1/2h^2	-2	1
y_1''	1/2h^2	2	-1
y_2''	1/6h^2	-3	3

	y_0	y_1	y_2	y_3	y_4	y_5
y_0'''''	1/120h^5	-15	45	-45	15	
y_1'''''	1/120h^5	15	-45	45	-15	
y_2'''''	1/720h^5	-105	315	-315	105	
y_3'''''	1/720h^5	105	-315	315	-105	
y_4'''''	1/5040h^5	-157	471	-471	157	
y_5'''''	1/5040h^5	157	-471	471	-157	

	y_0	y_1	y_2	y_3	y_4	y_5
y_0''''''	1/720h^6	-12	36	-36	12	
y_1''''''	1/720h^6	12	-36	36	-12	
y_2''''''	1/2520h^6	-105	315	-315	105	
y_3''''''	1/2520h^6	105	-315	315	-105	
y_4''''''	1/15120h^6	-175	525	-525	175	
y_5''''''	1/15120h^6	175	-525	525	-175	

	y_0	y_1	y_2	y_3	y_4	y_5	y_6
y_0'''''''	1/5040h^7	-14	42	-42	14		
y_1'''''''	1/5040h^7	14	-42	42	-14		
y_2'''''''	1/30240h^7	-105	315	-315	105		
y_3'''''''	1/30240h^7	105	-315	315	-105		
y_4'''''''	1/120960h^7	-175	525	-525	175		
y_5'''''''	1/120960h^7	175	-525	525	-175		
y_6'''''''	1/362880h^7	-21	63	-63	21		

	y_0	y_1	y_2	y_3	y_4	y_5	y_6
y_0''''''''	1/2520h^8	-15	45	-45	15		
y_1''''''''	1/2520h^8	15	-45	45	-15		
y_2''''''''	1/15120h^8	-105	315	-315	105		
y_3''''''''	1/15120h^8	105	-315	315	-105		
y_4''''''''	1/50400h^8	-175	525	-525	175		
y_5''''''''	1/50400h^8	175	-525	525	-175		
y_6''''''''	1/151200h^8	-21	63	-63	21		



Nachtrag zu ⑤

Zu den abgemagerten Treibern für V24 hat Holger Petersen richtig angemerkt, daß eine Diode sinnvoll ist den Transistor der zu schützen, da die Ausgänge falsch initialisiert sein können (Bild 1).

In den Unterlagen von Klaus Schemmert findet sich noch eine elegantere Schaltung für Versorgung aus 5V (Bild 2).

Manchmal ist optische Isolierung erwünscht (Bild 3). Umfangreiche Außenbeschaltung muß die Langsamkeit der Optokoppler ausgleichen. Dabei kann man die Arbeitswiderstände nicht beliebig niederohmig machen, weil die Current Transfer Ratio üblicher Optokoppler oft nur 50% beträgt.

Bild 4 schließlich zeigt das Prinzipschaltbild einer pragmatischen Testschaltung für die Opto-V24. Es handelt sich um einen Ringoszillator. Links ist ein MAX232 in der Schleife der das empfangene Signal sofort wieder sendet und die Hilfsspannung an DTR/DSR erzeugt (Bild 5). Das ist sinnvoll, weil der MAX232 der in der Praxis anzunehmende schlimmste Fall ist. Sein typischer Eingangswiderstand von 5k Ohm wird durch die externen Lastwiderstände auf 3k gesenkt.

Rechts, auf der „Controller“-Seite ist das erste XOR als Puffer, das zweite als Inverter beschaltet (Bild 4). Mit den beiden anderen Gates kann man bequem die Verzögerung durch die V24-Treiber messen (Bild 6). Sie wird durch zwei high-Pulse dargestellt. Deren absolute Dauer ist nicht so wichtig, da reine Laufzeit die Übertragung nicht stört. Erwünscht ist vielmehr, daß beide Pulse gleich lang sind, also die Signalform sich nicht ändert. Bei der Opto-V24 sind diese Werte 6/8 bzw. 3/6 usec. Wobei darin die typischen 0,5usec des MAX232 enthalten sind. Da bei 19200 Baud ein Bit 52usec dauert, ist der Jitter von 2-3usec akzeptabel.

Bild 1: mit Schutzdiode

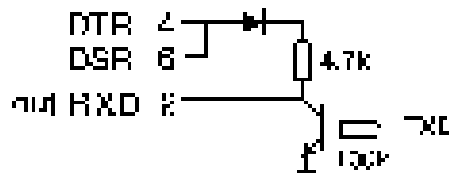


Bild 2: ohne Hilfsversorgung

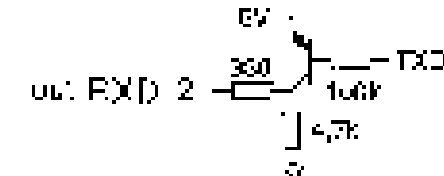


Bild 3: optisch isoliert

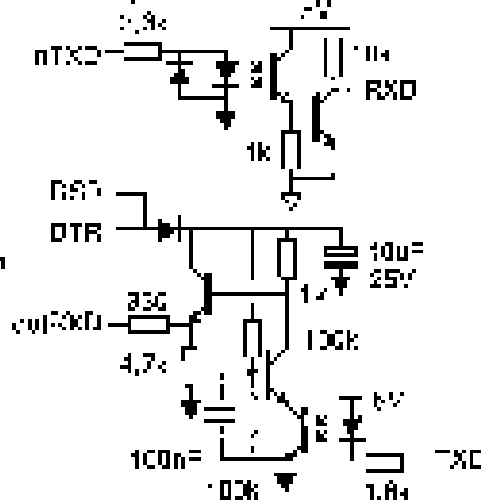


Bild 4: Blockschaltbild Testschaltung

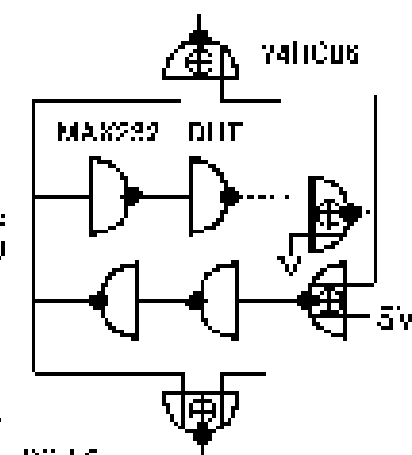
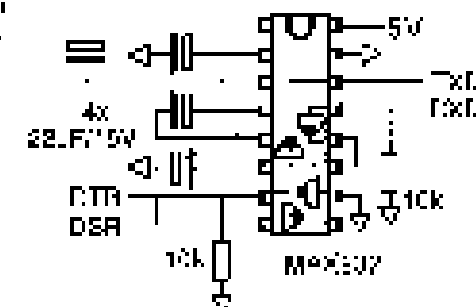


Bild 5: Linke Seite der Testschaltung: MAX232 koppelt Signal zurück



Nachtrag zu ④

Zu den Zufallszahlengeneratoren nach dem Kongruenzverfahren ist ein recht viel mehr wertiges Generationsverfahren der SAHLE-Formelgenauigkeit folgende von Apple gefundene

$$1 \times 10^8 - 2$$

$$x = (x^2 \cdot a) \bmod (x^2 - 1)$$

