03 – Codierungen

Technische Grundlagen der Informatik

Automation Systems Group E183-1

Institute of Computer Aided Automation
Vienna University of Technology

email: tgi@auto.tuwien.ac.at

Inhalt

- Zeichencodierungen
  - ASCII, Unicode, UTF, ISO-Latin-1
- Sonstige gängige Codierungen
- Codierungstheorie, Informationstheorie
  - Grundlagen
  - Datenübertragung
  - Arten von Codes
- Fehlererkennende und fehlerkorrigierende Codes
  - Hamming-Distanz
  - Prüfstellen-Codes
  - Polynomcodes

Zeichencodierungen

- ASCII
- Unicode
- UTF
- ISO-Latin-1 (ISO 8859-1)

ASCII

- Binärcodierung von Zeichen
- American Standard Code for Information Interchange
  - vom American National Standards Institute (ANSI) festgelegt
  - (in der ursprünglichen Version) 7 Bits zur Codierung
  - es lassen sich also $2^7 = 128$ Zeichen darstellen

- Seit 1963: US-ASCII-Code
  - Einführung von Kleinbuchstaben
  - 8. Bit als Prüfbit
  - Beschränkung auf 128 Zeichen – Probleme bei Sonderzeichen!

US-ASCII-Code-Tabelle (nicht prüfungsrelevant)

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Quelle: ascii-table.com

- Erst Spalte, dann Zeile auslesen
- z.B. Q ... $(51)16$

Unicode

- 1991: Vereinheitlichung aller Zeichen im **Unicode 1.0**
  - 16-Bit-Code
  - Darstellung von 2^{16} = 65 536 Zeichen
  - US-ASCII-Code gleich zu Beginn des Coderaums

- internationaler Standard
  - zur Zusammenfassung bekannter Textzeichen in einem Zeichensatz
  - ursprünglich als 2-Byte-Code konzipiert
  - inzwischen auch 4-Byte-Variante
    - mit der Möglichkeit, die Codierung weiterer Zeichen zu standardisieren

- Beispiel: "☎"
  - Hex: 260E
  - Binär: 0010 0110 0000 1110

- http://unicode-table.com/de/

UTF-Codierung

- Unicode Transformation Format
  - Verfahren zur Abbildung von Unicode-Zeichen auf Byte-Folgen
  - De-facto-Standard für Zeichencodierung im Internet (>80%)

- UTF-8
  - Je nach Zeichen 1, 2, 3 oder 4 Bytes (variable Länge!)
  - ausgefeiltes Verfahren, um Texte (aus einem lateinischen Alphabet) mit möglichst wenig Bytes darzustellen
  - Mit 1 Byte: Alle ASCII-Zeichen wie in erweiterter ASCII-Code-Tabelle
    - Erste 128 Zeichen des Unicode ident mit ASCII
  - Mit 2 Byte: andere Sonderzeichen wie Umlaute
  - Mit bis zu 4 Byte: kyrillische, fernöstliche und afrikanische Sprachen

  - Windows: UTF-16

UTF-Codierung (nicht prüfungsrelevant)

Unicode-Bereich (hexadezimal)	**UTF-8**-Codierung (binär)	Bemerkungen	Möglichkeiten (theoretisch)	
0000 0000 – 0000 007F	0xxxxxxx	In diesem Bereich (128 Zeichen) entspricht UTF-8 genau dem **US-ASCII**-Code: Das höchste Bit ist *0*, die restliche 7-Bit-Kombination ist das ASCII-Zeichen.	2^7	128
0000 0080 – 0000 07FF	110xxxxx 10xxxxxx	Das erste Byte beginnt immer mit 11, die folgenden Bytes mit 10. Die xxxxx stehen für die Bits des Unicode-Zeichenwerts. Dabei wird das niederwertigste Bit des Zeichenwerts auf das rechte x im letzten Byte abgebildet, die höherwertigen Bits fortschreitend *von rechts nach links*. Die Anzahl der Einsen vor der ersten 0 im ersten Byte ist gleich der Gesamtzahl der Bytes für das Zeichen. (In Klammern jeweils die theoretisch maximal möglichen.)	$2^{11} - 2^7$ (2^{11})	1920 (2048)
0000 0800 – 0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx		$2^{16} - 2^{11}$ (2^{16})	63.488 (65.536)
0001 0000 – 0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx		2^{20} (2^{21})	1.048.576 (2.097.152)

Quelle: Wikipedia

ISO-Latin-1 (ISO 8859-1)

- Zweithäufigster Zeichensatz im Web (<10%)
- versucht, möglichst viele Zeichen westeuropäischer Sprachen abzudecken
- 8-Bit Zeichencodierung
- Führende 0 + ASCII
- Führende 1 + 96 weitere darstellbare Zeichen

- Ähnlich: Windows-1252 Westeuropäisch

- Unterschiede zu Unicode und Windows-1252
  - ein paar Sonderzeichen
  - Steuerzeichen
  - Kein Euro-Zeichen €

Tabelle für ISO/IEC 8859-1 (nicht prüfungsrelevant)

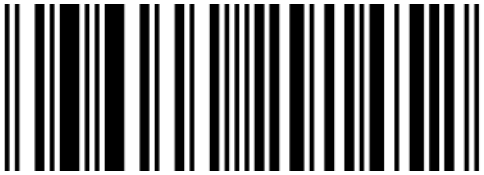
Code	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	*nicht belegt*															
1…																
2…	*SP*	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3…	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4…	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5…	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6…	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7…	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8…	*nicht belegt*															
9…																
A…	*NBSP*	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	*SHY*	®	¯
B…	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C…	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D…	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E…	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F…	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Quelle: Wikipedia

SP … space, NBSP … non-breaking space, SHY … soft hyphen

Sonstige gängige Codierungen

- Barcodes



- Hier nicht betrachtet
  - Audiosignale
  - Bilddaten
  - Video

Barcode – ISO/IEC 15420

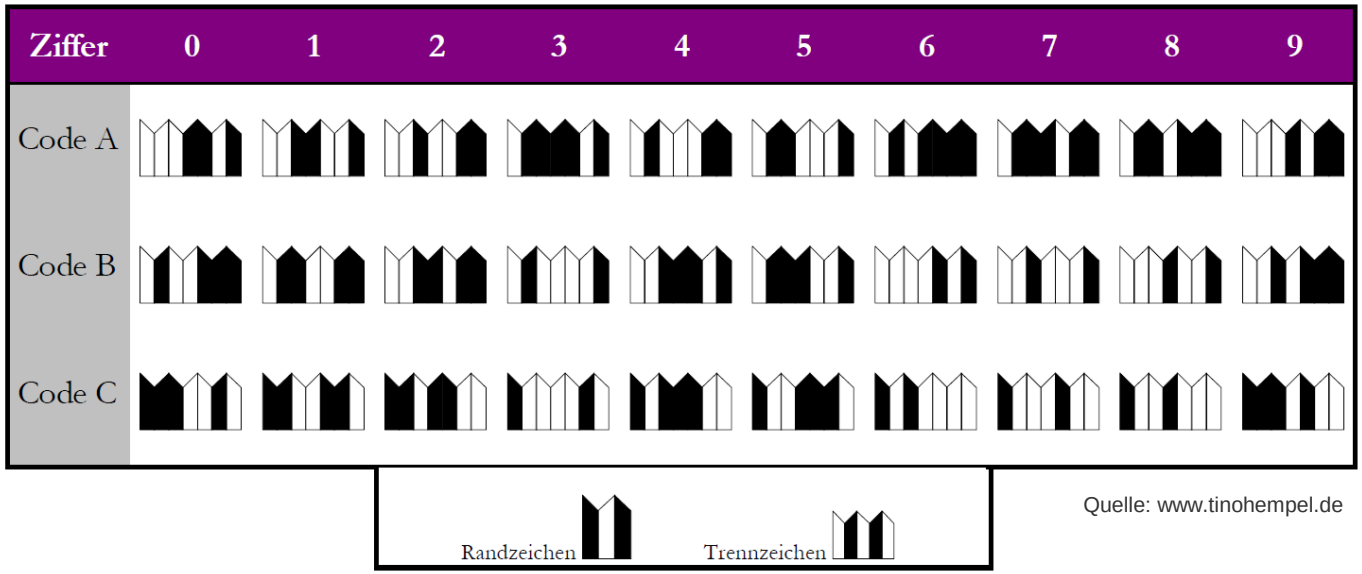
- Strichcode, Balkencode, Streifencode oder Barcode
  - opto-elektronisch lesbare Schrift
  - aus verschieden breiten, parallelen Strichen und Lücken
- maschinell eingelesen und elektronisch weiterverarbeitet
  - mit optischen Lesegeräten, wie z. B.
  - Barcodelesegeräten (Scanner) oder Kameras

- Patent aus 1952, in Wien seit 1979 verwendet

- Zeichenvorrat: 0…9
- EAN-13 codiert 12 Stellen plus eine Prüfziffer
  - meist im Klartext darunter
  - 1. implizite Ziffer links daneben

- ISBN und ISSN werden im EAN-13-Code codiert

Darstellung von EAN



Quelle: www.tinohempel.de

- Einzelne Ziffer codiert mit je
  - zwei dunklen und zwei hellen Streifen
  - unterschiedlicher Breite
- insgesamt 32 Symbole
  - Ziffern 0…9 mit jeweils 3 Symbolen (Codierung A, B, C),
  - ein Randsymbol als erstes und letztes Zeichen
  - ein Trennsymbol in der Mitte

Aufbau von EAN

- Prüfziffer *p* (13. Ziffer) errechnet mit

$$(\underbrace{z_1}_{\text{implizit}} + \underbrace{3z_2 + z_3 + 3z_4 + z_5 + 3z_6 + z_7}_{\text{A, B}} + \underbrace{3z_8 + z_9 + 3z_{10} + z_{11} + 3z_{12} + p}_{\text{C}}) \equiv 0 \; mod \; 10$$

- Rechter Teil: Symbole aus Satz C
- Linker Teil: Symbole aus A und B (abhängig von erster Ziffer)
  - Erste Ziffer implizit

1. Ziffer	Codemuster für linke Seite
0	AAAAAA
1	AABABB
2	AABBAB
3	AABBBA
4	ABAABB
5	ABBAAB
6	ABBBAA
7	ABABAB
8	ABABBA
9	ABBABA

Quelle: www.tinohempel.de

Beispiel EAN

$$z_1 + 3z_2 + z_3 + 3z_4 + z_5 + 3z_6 + z_7 + 3z_8 + z_9 + 3z_{10} + z_{11} + 3z_{12} + p \equiv 0 \; mod \; 10$$

$$1 + 3 \cdot 2 + 3 + 3 \cdot 4 + 5 + 3 \cdot 6 + 7 + 3 \cdot 8 + 9 + 3 \cdot 0 + 1 + 3 \cdot 2 + p \equiv 0 \; mod \; 10$$

$$1 + 6 + 3 + 12 + 5 + 18 + 7 + 24 + 9 + 0 + 1 + 6 + p \equiv 0 \; mod \; 10$$

$$92 + p \equiv 0 \; mod \; 10$$

$$p = 8$$

Codierungstheorie, Informationstheorie

- Informationstheorie – Definition
- Codierungstheorie – Definition
- Datenübertragung mit mehrstufiger Codierung
- Arten von Codes

Informationstheorie

- mathematische Theorie aus Wahrscheinlichkeitstheorie und Statistik
- geht auf Claude Shannon zurück
- theoretische Betrachtung von Kommunikation
- beschäftigt sich mit
  - Information
    - Informationsgehalt eines Zeichens: minimale Anzahl von Bits zur Darstellung
  - Entropie
    - Je ungleichförmiger eine Nachricht, desto höher ihre Entropie
  - sowie Informationsübertragung, Datenkompression, Codierung und verwandten Themen
- Verwendung in
  - Mathematik, Informatik und Nachrichtentechnik

Shannon – Informationstheorie

- 1940er bis 1950er Jahren „Theorie der Datenübertragung“
  - verlustfreie Datenübertragung
- Ziel
  - Erkennung und Korrektur von Fehlern, die während der Übertragung entstanden sind
- Weg
  - Mitsenden von redundanten Daten
    also zusätzliche Daten, die keine Information tragen
- Definition von Information als „physikalische Größe“
  - mit einer Maß- bzw. Zähleinheit: Bit
- Ausdrücklicher Ausschluss von
  - semantischen und pragmatischen Aspekten der Information
  - Aussagen über den „Inhalt“ übertragener Nachrichten sowie deren Bedeutung für den Empfänger

- Frage: wie codieren?

Codierungstheorie – Definition

- Math. Theorie der fehlererkennenden und -korrigierenden Codes
- Ziel: Schutz vor Fehlern bei Übertragung oder Speicherung
- Beruht größtenteils auf Algebra, aber auch Methoden aus
  - Kombinatorik, Zahlentheorie, Geometrie
- Begründer: Golay und Hamming, ca. 1950

Codierungen – Ziele

- Je nach Ziel unterschiedliche Datenmodifikationen (Codierungen)
  - Codierungstheorie: Absicherung gegen Übertragungsfehler (durch erhöhte Redundanz)
  - Kryptographie: Absicherung gegen ungewollte Empfänger und Sender
  - Datenkompression: Reduktion der Datenmenge

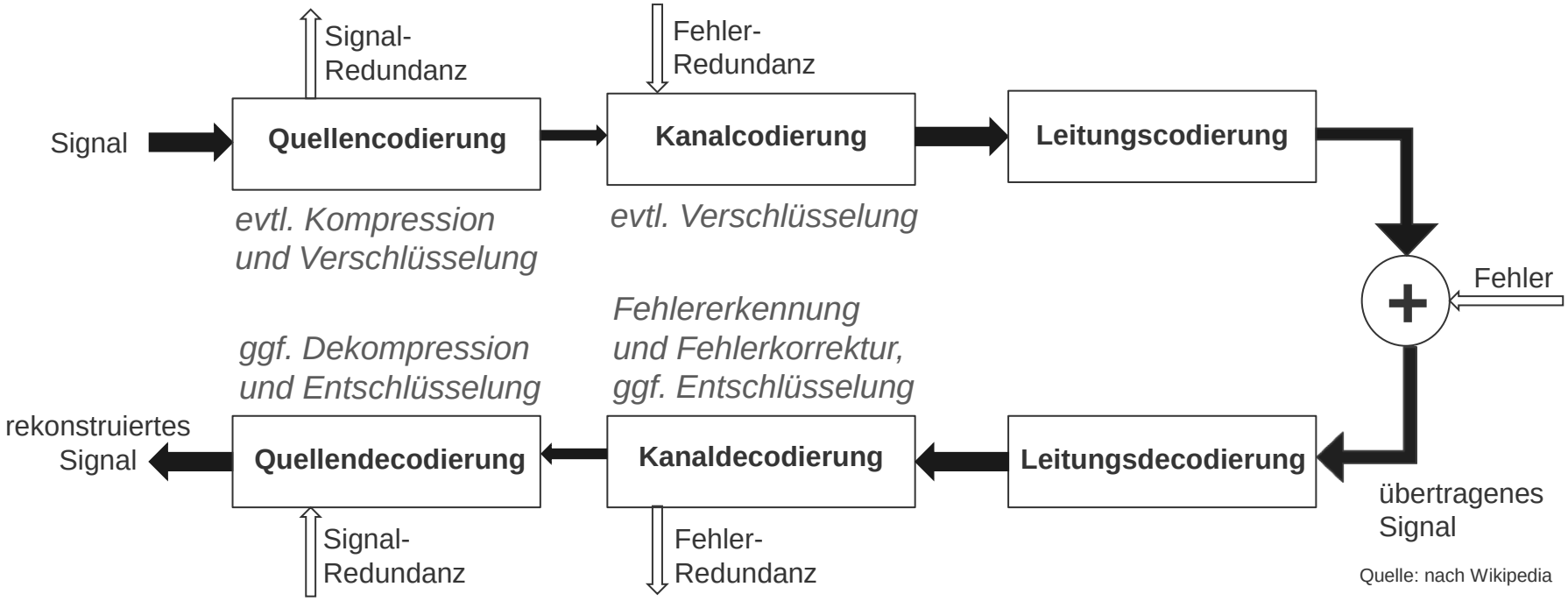
- Können miteinander kombiniert werden:
  - Nicht unüblich, dass Daten
    - komprimiert,
    - kryptographisch verschlüsselt und
    - gegen Übertragungsfehler codiert werden.

- Kompression hilfreich für statistische Gleichverteilung der Zeichen → bessere Absicherung gegen Übertragungsfehler

Codierung – Redundanz

- „Eine Informationseinheit ist dann redundant, wenn sie ohne Informationsverlust weggelassen werden kann.“ [wiki]
- „Redundant ist der Teil einer Nachricht, der keine Information enthält.“ [wiki]
- Gezielter Einsatz zur Fehlererkennung und -korrektur
- Steigerung der Qualität (weniger Fehler)
  auf Kosten der Quantität (niedrigere Nutzdatenrate)
- Stärke der Redundanz: anwendungsabhängig
  - Sicherheitskritische Systeme (viel)
  - Telefonie (wenig)

Datenübertragung mit mehrstufiger Codierung



Quelle: nach Wikipedia

Die Quellencodierung entfernt überflüssige Information einer Datenquelle (reduziert also die Redundanz in den Ausgangsdaten), was auch als Datenkompression bezeichnet wird. Eventuell wird auch verschlüsselt.

Danach fügt die Kanalcodierung zusätzliche Daten hinzu (erhöht also die Redundanz gezielt wieder), um mögliche Übertrags- bzw. Speicherfehler erkennen und korrigieren zu können.

Die Leitungscodierung passt das nun mehrfach codierte Signal durch eine weitere Umcodierung an das Übertragungsmedium an.

Bei der Decodierung wird in umgekehrter Reihenfolge das übertragene Signal so decodiert, dass das ursprünglich Signal möglichst gut rekonstruiert wird.

Quellencodierung

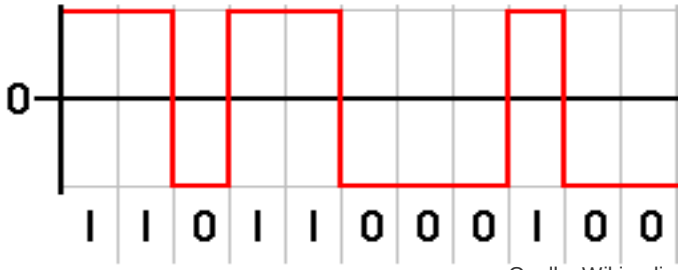
- Quellencodierung
  - Originaldaten werden binär codiert
  - Kompression üblich, Verschlüsselung möglich

- Kanalcodierung
  - Umcodierung der Binärdaten (evtl. komprimiert und verschlüsselt) d.h. Einfügen zusätzlicher Bits, um Übertragung sicherer zu gestalten (Hinzufügen von Redundanz)
  - Fehlerkorrigierende Codes üblich, Verschlüsselung möglich

- Leitungscodierung
  - Nochmaliges Umcodieren der Daten, um sie auf Übertragungsmedium zu optimieren
  - „Dabei werden bestimmte Pegelfolgen, etwa Lichtintensitäten auf Glasfasern oder Spannungen oder Ströme auf elektrischen Leitungen, binären Bitsequenzen im Datenstrom zugeordnet.“ [wiki Leitungscode]

Leitungscode NRZ

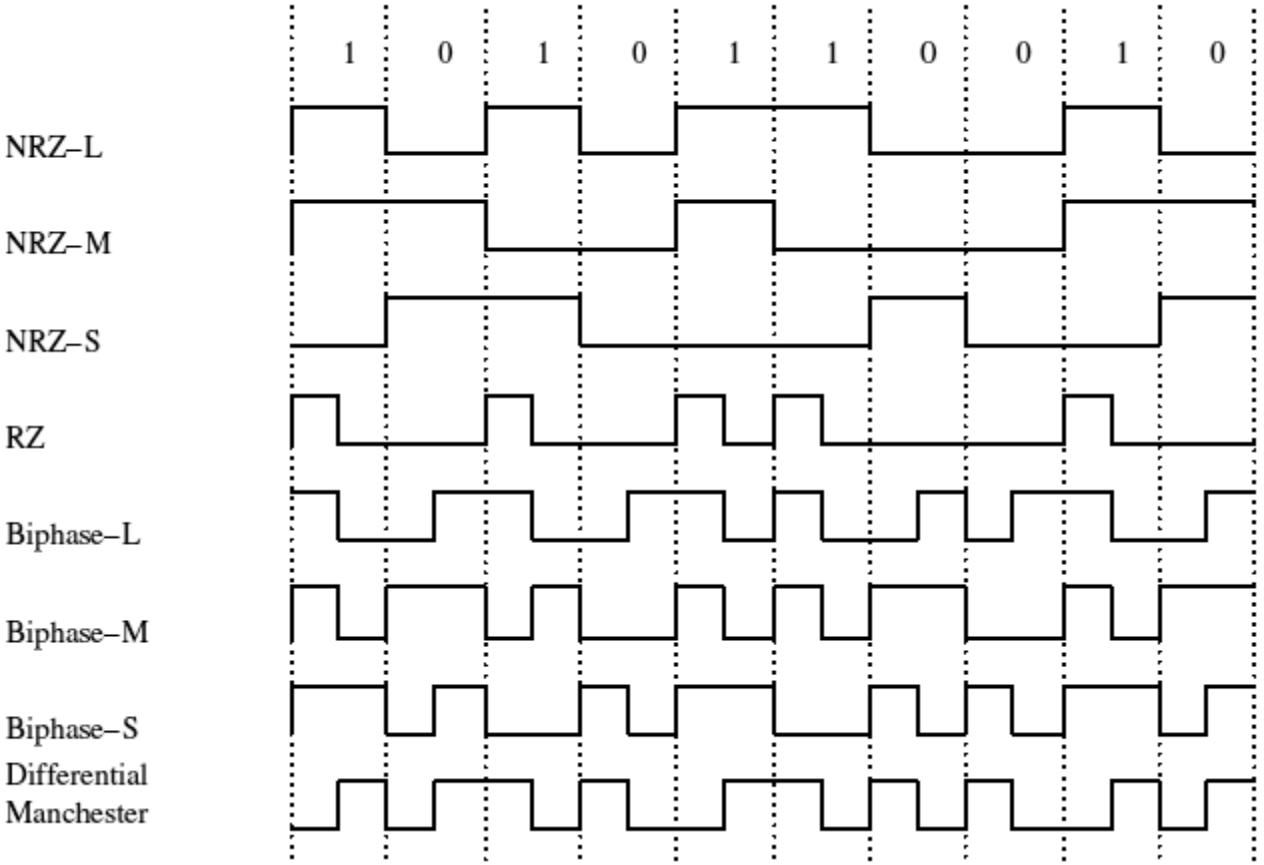
- Non Return to Zero (NRZ):
  - zwei Pegelzustände (high / low)
  - jeder Leitungszustand trägt Information



Quelle: Wikipedia

- Zuordnung je eines Leitungspegels für die logischen Zustände 0 und 1
  - ein Leitungspegel kann auch 0 Volt sein
- separates Taktsignal nötig
- Verwendung, wenn in Nutzdaten keine langen konstanten Folgen
  - z.B. ASCII-codierte Texte
  - Grenze für ‘lang’ vom Medium abhängig
- bei magnetischer Datenaufzeichnung problematisch
- verwendet u.a. beim CAN-Bus (Bus in Automatisierung)

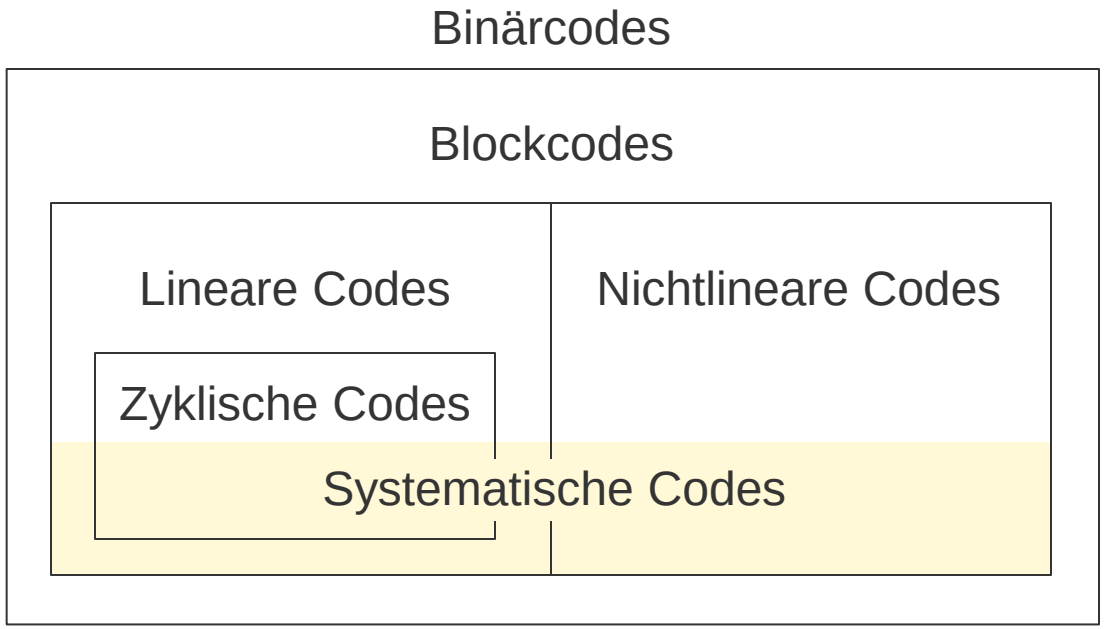
Leitungscode – Varianten von NRZ



Quelle: Wikibooks

- NRZ-L codiert 1 mit „high“, 0 mit „low“
- NRZ-M bewirkt einen Pegelwechsel bei 1
- NRZ-S bewirkt einen Pegelwechsel bei 0
- RZ: unipolarer Return-to-Zero-Code zweite Takthälfte immer „low“
- Biphase: für jedes Bit zwei Zustände
- D. Manchester: mind. eine Flanke pro Bit, Takt „mitcodiert“, geänderte Flanke codiert 1

Arten von Codes



- **Blockcodes**: konstante Codewortlänge
- **Lineare Codes**: Summe von Codewörtern ist auch ein gültiges Codewort (Modulo-Arithmetik!)
- **Zyklische Codes**: Durch bitweises Rotieren (Shiften) eines Codewortes entsteht wieder ein gültiges Codewort
- **Systematische Codes**: Codewörter der Länge n bestehen aus m Informationsbits gefolgt von $\mathrm{r} = (n - m)$ Prüfbits (Prüfstellen)

Linearer Code

- spezieller Blockcode
- Linear-Kombination von zwei Codewörtern ebenfalls Codewort
  - d.h. die Summe bzw. Vielfache
- Modulo2-Arithmetik! $1 + 1 = 0$
- Vorteil
  - Verwendung von Methoden der Linearen Algebra
  - ➢ einfach zu codieren und decodieren
- viele wichtige Codes sind linear
  - z.B. Hamming-Code
  - alle zyklischen Codes

Zyklischer Code

- wichtiger Kanalcode
- linearer Blockcode
- Anwendung bei
  - Übertragungskanälen
  - Datenspeichern

- Definition
  - Ein zyklischer Code C ist ein linearer Code, der zusätzlich folgende Eigenschaft hat:

    Ist $(a_0, a_1, \cdots, a_{n-1})$ ein Codewort von C,

    dann ist auch $(a_1, \cdots, a_{n-1}, a_0)$ ein Codewort von C.

    Daraus folgt, dass auch $(a_2, a_3, \cdots, a_{n-1}, a_0, a_1)$,

    $(a_3, a_4, \cdots, a_{n-1}, a_0, a_1, a_2)$ $\cdots$ $(a_{n-1}, a_0, a_1, \cdots, a_{n-2})$

    Codewörter von C sind.

Fehlererkennende und fehlerkorrigierende Codes

- Hamming-Distanz
- Prüfstellen-Codes
- Polynomcodes

Fehlererkennende und fehlerkorrigierende Codes

- Bei der Übermittlung von Information treten unweigerlich Störungen auf, z.B. bei Telefon

- Problem bei redundanzarmen Codierungen
  - bei „Umfallen“ eines Bits von 0 auf 1 oder von 1 auf 0
  - ➢ Inhalt der übertragenen Information zerstört oder geändert

Zwei Arten von Übertragungsfehlern:

- Einzelbit-Fehler
  - einzelne, unzusammenhängende Bits gestört
- Fehlerbündel (engl. burst)
  - mehrere aufeinanderfolgende Bits gestört

Gestörte Übertragung

- Gesucht: Maß für Störungsanfälligkeit
  „Wie sicher ist ein Code gegenüber Störungen?“

- z.B.: Alphabet {a | e | i | o} mit folgender Codierung

a	00
e	01
i	10
o	11

- Ein einziges Bit gestört → falsche Nachricht wird empfangen
  - zu übertragende Nachricht: „a“ (00)
  - falls erstes Bit gestört (10 statt 00) → „i“ empfangen
  - falls zweites Bit gestört (01 statt 00) → „e“ empfangen

Gestörte Übertragung

- Wählt man folgende Codierung, ist es durch Ändern eines einzigen Bits nicht mehr möglich, ein anderes gültiges Codewort zu erhalten

a	000
e	011
i	101
o	110

- zu übertragende Nachricht: „a“ (000)
  - falls erstes Bit gestört (100 statt 000) → ungültiges Codewort
  - falls zweites Bit gestört (010 statt 000) → ungültiges Codewort
  - falls drittes Bit gestört (001 statt 000) → ungültiges Codewort
- Fehlererkennung: Ja
  - ungültiges Codewort → Übertragungsfehler aufgetreten
- Fehlerkorrektur: Nein
  - z.B. 010 empfangen → sollte „a“ oder „e“ oder „o“ übertragen werden?

Hamming-Distanz

- An wie vielen Stellen unterscheiden sich zwei Codewörter?

a	00
e	01
i	10
o	11

- z.B.: a mit 00 und e mit 01 unterscheiden sich an einer Stelle
  → Hamming-Distanz $D = 1$

- Bestimmung der Hamming-Distanz für je zwei Codewörter
  - in einer Matrix
  - minimaler Wert legt Hamming-Abstand des Codes fest

Hamming-Distanz

- Erster Fall: Code mit $D = 1$

a	00
e	01
i	10
o	11

	a	e	i	o
a	-	1	1	2
e	1	-	2	1
i	1	2	-	1
o	2	1	1	-

- Zweiter Fall: Code mit $D = 2$

a	000
e	011
i	101
o	110

	a	e	i	o
a	-	2	2	2
e		-	2	2
i			-	2
o				-

Hamming-Distanz (Hamming-Abstand)

Quelle: Wikipedia

- Maß für die Unterschiedlichkeit von Zeichenketten
- Bei Binärcodes
  - Vergleich durch eine XOR-Operation (Modulo-Addition)
  - Abzählen der resultierenden Einsen
  - Anzahl der Einsen im Vergleich ergibt Hamming-Distanz
- Hamming-Abstand eines Codes
  - Minimum aller Abstände zwischen Wörtern innerhalb des Codes
- Bei Codes mit Hamming-Abstand D sind alle
  - $(D-1)$ Bit-Fehler erkennbar und
  - k Bit-Fehler, $k < \frac{D}{2}$ korrigierbar

Parity-Bit (Paritätsbit)

- Erhöhung der Hamming-Distanz eines Codes von 1 auf 2
- dient der Erkennung fehlerhaft übertragener Informationswörter
- erfolgt durch Anhängen eines Parity-Bits
  - Jedem Codewort wird ein Bit hinzugefügt
  - Even parity: Anzahl der Einsen in einem Codewort gerade
    → Parity-Bit wird 0
  - Odd parity: Anzahl der Einsen in einem Codewort ungerade
    → Parity-Bit wird 0
- Mehrere Paritätsbits?
  - (mp)p

Prüfstellen-Codes (systematische Codes)

- Ein Paritätsbit
- Mehrere Paritätsbits
  - je ein Paritätsbit für unterschiedliche Teile des Informationswortes
    z.B. **Hamming-Code**
  - mehrdimensionale Paritäts-Verfahren
    z.B. Kreuz- oder Blockparität
- Prüfsumme
  - z.B.: EAN (nicht binär!)
  - Polynomcodes
    - Zyklische Codes, z.B. **CRC-Codes**

Polynomcodes

- sind Codes mit mehreren Prüfbits
- Bitfolgen als Polynome interpretiert,
  deren Koeffizienten nur aus 0 und 1 bestehen
- Interpretation einer Bitfolge $(a_{k-1} \cdots a_1 a_0)$ der Länge k
  als Koeffizienten eines Polynoms vom Grad $k-1$

$$a_{k-1}x^{k-1} + \cdots + a_1 x^1 + a_0 x^0$$

- z.B.: Bitfolge „110001“ entspricht dem Polynom $x^5 + x^4 + x^0$

Polynomcodes – Idee

- Generator-Polynom $G(x)$ zur Berechnung der Prüfsumme
- Anhängen dieser Prüfsumme an die Nachricht, d.h. an das Message-Polynom $M(x)$
- Übertragen beider Teile (Message und Prüfsumme) als ein Codewort
- Prüfsumme so wählen, dass das übertragene Wort durch $G(x)$ ohne Rest teilbar ist
- Ergibt das erhaltene Wort bei Division durch $G(x)$ einen Rest ungleich 0, so liegt ein Übertragungsfehler vor

- „Grad“ eines Polynoms → höchste vorkommende Potenz von x
  - z.B. $(x^5 + x^4 + 1)$ → Polynom fünften Grades
- Achtung! Rechnungen mit diesen Polynomen immer modulo 2

$$1 + 1 = 0 \qquad -1 = +1$$

Rechnen mit Polynomen modulo 2

- Beispiel: Addition

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

$$\text{da } 1 \cdot x^5 + 1 \cdot x^5 = 0 \cdot x^5 = 0$$

- Beispiel: Multiplikation

$$(x + 1) \cdot (x + 1) = (x^2 + 1)$$

$$\text{da } 1 \cdot x + 1 \cdot x = 0 \cdot x = 0$$

Rechnen mit Polynomen modulo 2

- Beispiel: Division

$(x^5 + x^4 + 0*x^3 + 0*x^2 + 0*x + 1) : (x^2 + 1) = x^3 + x^2 + x + 1$

$$
\begin{array}{rl}
x^5 + x^3 & \\
\hline
x^4 + x^3 + 0 & \\
x^4 + x^2 & \\
\hline
x^3 + x^2 + 0 & \\
x^3 + x & \\
\hline
x^2 + x + 1 & \\
x^2 + 1 & \\
\hline
x &
\end{array}
$$

Rest: $R(x) = x$

Algorithmus zur Berechnung der Prüfsumme

1. Sei r der Grad von $G(x)$ und m der Grad von $M(x)$

   Man hängt r Nullen am Ende des Codewortes an,
   das Codewort hat nun $m + r$ Bits und
   entspricht dem Polynom $x^r \cdot M(x)$

2. Man dividiert $x^r \cdot M(x)$ durch $G(x)$,
   wobei $1 + 1 = 0$ berücksichtigt wird

3. Man subtrahiert den bei der Division erhaltenen Rest $R(x)$ vom Polynom $x^r \cdot M(x)$. Das Resultat $T(x)$ dieser Subtraktion wird übertragen.

$$T(x) = x^r \cdot M(x) - R(x)$$

Polynomcode – Beispiel

- Sei $G(x) = x^3 + x^2 + 1$; Nachrichten-Wort „0110“
- Gesucht: zu übertragende Bitfolge

- $M(x) = 0 + x^2 + x + 0 = x^2 + x$
- r … Grad von $G(x)$, hier $r = 3$
- Das Polynom $x^r \cdot M(x) = x^3 \cdot (x^2 + x) = x^5 + x^4$
  muss nun durch $G(x)$ dividiert werden

$(x^5 + x^4) : (x^3 + x^2 + 1) = x^2$

$\underline{x^5 + x^4 + x^2}$

x^2 Rest

- Der Rest $R(x)$ muss von $x^r \cdot M(x)$ abgezogen werden:
  $T(x) = x^r \cdot M(x) - R(x) = x^5 + x^4 - x^2 = x^5 + x^4 + x^2$

- → zu übertragende Bitfolge: „0110**100**“

Übertragungsfehler – Erkennung

- Angenommen bei der Übertragung tritt eine Störung auf. Es kommt also nicht das Polynom $T(x)$ an, sondern das gestörte Polynom $T(x) + E(x)$, wobei $E(x)$ das Error-Polynom sei.

- Bei der Decodierung wird das empfangene Polynom durch $G(x)$ dividiert

$$(T(x) + E(x)) / G(x)$$

- Da $T(x)$ ohne Rest durch $G(x)$ dividierbar ist, bleibt als Rest nur der Rest von $E(x)/G(x)$

- Alle Fehler die durch $G(x)$ teilbar sind können daher nicht, alle anderen sehr wohl, erkannt werden.

Übertragungsfehler – Beispiel

- Bei der Übertragung wird das Wort „0110100“ gestört, es wird stattdessen „1110100“ empfangen
- Diese Bitfolge entspricht dem Polynom $x^6 + x^5 + x^4 + x^2$
- Bei der Division dieses Polynoms durch $G(x)$ ergibt sich als Resultat $x^3 + x$ und als Rest $x^2 + x$
- Der Rest ist ungleich 0, der Übertragungsfehler wird somit erkannt
- Zur Fehlerkorrektur kann ggfs. der Rest in einer Tabelle nachgeschlagen werden

CRC-Codes (Cyclic Redundancy Check)

- zyklische Polynomcodes
- 1961 von Peterson entwickelt
- dienen der Fehlererkennung
- einfach in Hardware zu implementieren
- einfach zu analysieren
- gut für Fehlererkennung bei verrauschten Kanälen
  - z.B. Telekommunikation, Netzwerke

CRC-Standard-Polynome

- Generator-Polynome, die zum Standard erhoben wurden:
  - $CRC - 12 = x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
  - $CRC - 16 = x^{16} + x^{15} + x^2 + 1$
  - $CRC - CCITT = x^{16} + x^{12} + x^5 + 1$

  - CRC-12 wird für Bitfolgen der Länge 6,
    die beiden anderen werden für Folgen der Länge 8 verwendet

- Aus wie vielen Bits besteht ein Codewort bei CRC-12?
  - 6 Datenbits
  - 12 Absicherungsbits
  - → 18 Bits insgesamt

Fehlerkorrigierende Codes

- Beispiel: Code aus vier Codewörtern
  0000000000, 0000011111, 1111100000 und 1111111111.

- Hamming-Distanz $D = 5 \rightarrow$ max. zwei Fehler können korrigiert werden

- Empfangene Nachricht: 0000000111
  - wenn höchstens zwei Fehler aufgetreten sind

    → Original 0000011111
  - Wenn drei Fehler aufgetreten sind:

    Original 0000000000 oder 0000011111?

    → Fehler kann nicht korrigiert werden

Fehlerkorrigierende Codes – Anzahl der Prüfbits

- n Codebits, davon m Datenbits, r Prüfbits, $n = m + r$
- Gewünscht: Korrektur von einem fehlerhaften Bit

- Wie viele Prüfbits r braucht man mindestens?

$m + r = n$ Codebits

m Datenbits	r Prüfbits

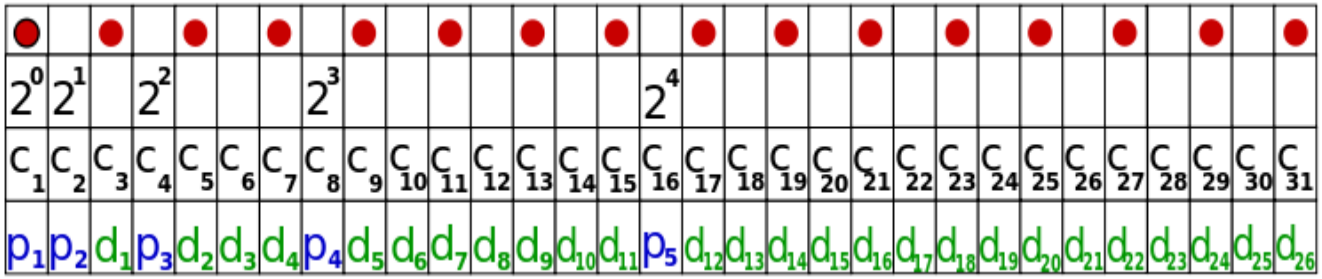
- Anzahl der möglichen Bitmuster in den Prüfbits: 2^r
- Um einen Fehler an allen n Stellen korrigieren zu können, müssen in den r Prüfbits zumindest folgende Bitmuster codiert sein:
  - 1 Bitmuster für fehlerfrei
  - je 1 Bitmuster für jede mögliche Fehlerposition, d.h. n weitere Bitmuster

➢ Also muss r erfüllen: $2^r \geq n + 1 = m + r + 1$

Hamming-Code

- von R. Hamming entwickelt
- linearer fehlerkorrigierender Blockcode
- Verwendung mehrerer Paritätsbits
  - Diese Bits ergänzen jeweils unterschiedlich gewählte Gruppen von den die Information tragenden Nutzdatenbits.
  - Durch eine geschickte Wahl der Gruppierung ist nicht nur eine Fehlererkennung, sondern auch eine Fehlerkorrektur der übertragenen Datenbits möglich.
- Hamming-Abstand $D = 3$
  → kann bis zu zwei Bitfehler in einem Codewort erkennen
  → kann einen Bitfehler korrigieren

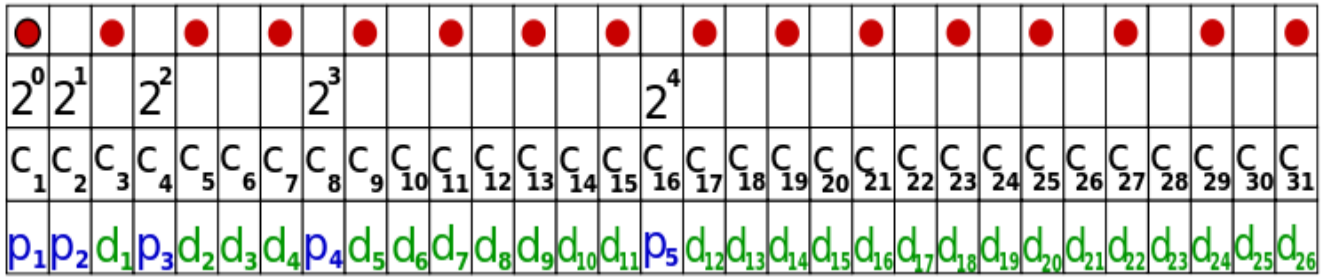
Hamming-Code



Quelle: Wikipedia

- Die Bits des Codewortes werden mit 1 beginnend von links nach rechts durchnummeriert: $c_1, c_2, c_3, \cdots$
- Jene Bits, die Potenzen von 2 sind, also 1, 2, 4, 8, 16, usw. sind Prüfbits: $p_1, p_2, p_3, \cdots$
- Die restlichen Bits ($c_3, c_5, c_6, c_7, c_9, \cdots$) sind mit den informationstragenden Datenbits gefüllt: $d_1, d_2, d_3, \cdots$
- Jedes Prüfbit ist ein Parity-Bit für eine bestimmte Menge von Bits
- Ein Bit kann in die Berechnung verschiedener Prüfbits involviert sein

Hamming-Code



Quelle: Wikipedia

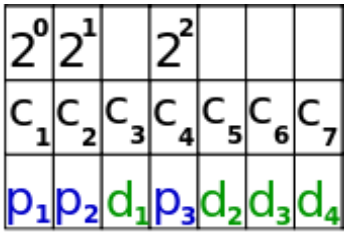
- Feststellen, zu welchen Prüfbits das Codebit mit Index k beiträgt: k als Summe von Zweierpotenzen darstellen
- z.B. $11 = 8 + 2 + 1$ oder $29 = 16 + 8 + 4 + 1$
- Ein Codebit liefert zu all jenen Prüfbits einen Beitrag, deren Index (als Codebit) in dieser Darstellung vorkommt.
- Codebit 11 liefert Beitrag zu Prüfbits an den Stellen 1, 2 und 8

Hamming-Code – Prüfung

- Wenn ein bestimmtes Wort empfangen wird, setzt man einen Korrekturindikator auf 0.
- Dann kontrolliert man jedes Prüfbit c_k $(k = 1, 2, 4, 8, \cdots)$, um zu sehen, ob der im empfangenen Codewort stehende Wert mit dem neu berechneten Prüfwert übereinstimmt.
- Falls das nicht der Fall ist, addiert man k zum Korrekturindikator. Ist dieser Indikator nach Bearbeitung aller Prüfbits gleich 0, so akzeptiert man das Wort als korrektes Codewort.
- Andernfalls enthält der Indikator die Nummer des gestörten Bits, das somit leicht korrigiert werden kann.
- z.B. die Prüfbits c_1, c_2 und c_8 waren nicht richtig, so muss das Bit 11 korrigiert (invertiert) werden, da $1 + 2 + 8 = 11$.

Hamming-Code – Beispiel

- Hamming-Code mit vier Datenbits und drei Prüfbits



Quelle: Wikipedia

- Datenbits c_3, c_5, c_6, c_7 und Prüfbits p_1, p_2, p_3

$$p_1 = c_1 = (c_3 + c_5 + c_7)\, mod\ 2 = c_3 \otimes c_5 \otimes c_7$$
$$p_2 = c_2 = (c_3 + c_6 + c_7)\, mod\ 2 = c_3 \otimes c_6 \otimes c_7$$
$$p_3 = c_4 = (c_5 + c_6 + c_7)\, mod\ 2 = c_5 \otimes c_6 \otimes c_7$$

$$0 \otimes 0 = 0, \qquad 1 \otimes 1 = 0$$
$$0 \otimes 1 = 1, \qquad 0 \otimes 0 = 1$$

$\otimes$ … XOR

Hamming-Code – Beispiel

- Datenwort: „0110“
- Berechne Prüfbits:

$$p_1 = c_1 = (c_3 + c_5 + c_7)\, mod\ 2 = c_3 \otimes c_5 \otimes c_7$$
$$p_2 = c_2 = (c_3 + c_6 + c_7)\, mod\ 2 = c_3 \otimes c_6 \otimes c_7$$
$$p_3 = c_4 = (c_5 + c_6 + c_7)\, mod\ 2 = c_5 \otimes c_6 \otimes c_7$$

$$p_1 = 0 \otimes 1 \otimes 0 = 1$$
$$p_2 = 0 \otimes 1 \otimes 0 = 1$$
$$p_3 = 1 \otimes 1 \otimes 0 = 0$$

Zu übertragendes Wort: **11**0**0**110

- Angenommen das Bit c_5 wird bei der Übertragung gestört und daher das Wort 1100**0**10 empfangen

Hamming-Code – Beispiel

Jetzt sehen die Berechnungen beim Empfang folgendermaßen aus:

$$p_1 = c_1 = (c_3 + c_5 + c_7)\, mod\ 2 = c_3 \otimes c_5 \otimes c_7$$
$$p_2 = c_2 = (c_3 + c_6 + c_7)\, mod\ 2 = c_3 \otimes c_6 \otimes c_7$$
$$p_3 = c_4 = (c_5 + c_6 + c_7)\, mod\ 2 = c_5 \otimes c_6 \otimes c_7$$

Die Berechnungen ergeben:

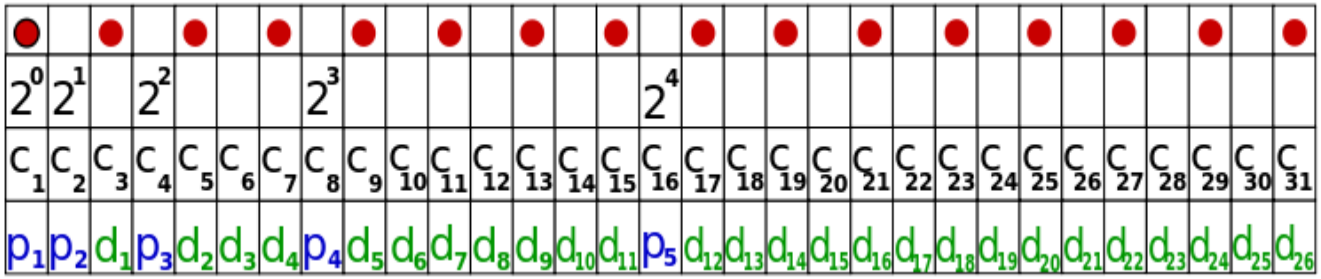
$$p_1 = 0 \otimes 1 \otimes 0 = 1 \neq 0$$
$$p_2 = 0 \otimes 1 \otimes 0 = 1 = 1$$
$$p_3 = 1 \otimes 1 \otimes 0 = 0 \neq 1$$

- Da die Berechnungen für die Prüfbits $p_1 = c_1$ und $p_3 = c_4$ einen anderen Wert ergeben als übertragen wurde, weiß man, dass der Korrekturindikator den Wert $5 = (1 + 4)$ hat und somit das gestörte Bit c_5 ist. (Unter der Annahme, dass nur ein Bit gestört wurde.)

Hamming-Code – Mehrfachfehler

- Auch beim Auftreten von mehreren gestörten Stellen kann der Indikator den Wert 0 haben oder aber einen Wert, der größer ist als die Länge der Codewörter.
- Falls der letztere Fall eintritt kann man also mit Sicherheit auf das Vorhandensein mehrerer gestörter Stellen schließen.
- Der hier vorgestellte Hamming-Code kann nur einfache Fehler korrigieren.
- Für Fehlerbündel kann man eine Folge von k aufeinanderfolgenden Codewörtern in Form einer Matrix anordnen, und zwar ein Codewort pro Zeile.
- Prüfsummen werden dann für Zeilen und Spalten gebildet, gesendet wird zeilenweise.

Hamming-Code – Gleichungen für Prüfbits



Quelle: Wikipedia

- Das Paritätsbit p_i wird über alle Stellen c_j des Codeworts berechnet, in denen an der i-ten Stelle der Binärcodierung des Index j eine logische 1 steht (außer diese Stelle ist ein Prüfbit)

- z.B. Datenwort $d_1\ d_2\ d_3\ d_4\ d_5\ d_6\ d_7\ d_8$
  - 8 Datenbits → 4 Prüfbits, also 12 Bit insgesamt
  - p_1 … immer die ungeraden Bits → $c_1, c_3, c_5, \ldots, c_{11}$
  - p_2 … jene c_j, wo in der orangen Spalte eine 1 gesetzt ist ($i = 2$) außer c_2 selbst → $c_3, c_6, c_7, c_{10}, c_{11}$
  - $p_3 \rightarrow c_5, c_6, c_7, c_{12}$ (Achtung c_4 nicht)
  - $p_4 \rightarrow c_9, c_{10}, c_{11}, c_{12}$ (Achtung c_8 nicht)

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

Zusammenfassung

- mehrfache Codierung
  - Quelle, Kanal, Leitung
  - Codierung, Verschlüsselung, Kompression

Zugang zur Codierung

- mathematische Grundlagen
  - um gewünschte Eigenschaften eines Codes zu gewährleisten

- technische Implementierung
  - Anpassung an das verwendete physikalische Medium
    - Speichermedium: optisch, magnetisch, elektrisch
    - Übertragungsmedium: elektrische Leitung, Funk