



## 1. Übung zur Vorlesung Rechnernetze (Musterlösung)

### Aufgabe 1: Quellcodierung und Leitungscodierung (2 Punkte, 1+1)

- Erklären Sie die Begriffe Quellcodierung und Leitungscodierung!
- Erläutern Sie, welchen Einfluss Quellcodierung und Leitungscodierung auf die Übertragungsdauer einer Nachricht (z.B. Text oder Bild) über einen Kanal mit fester Baudrate haben!

*Lösungsvorschlag:*

- Quellcodierung definiert das Format, in dem Daten im Rechner oder auf einem Speichermedium dargestellt werden, z.B. ASCII-Zeichensatz für Text. Darüber hinaus kann die Quellcodierung für eine Komprimierung der Daten sorgen, z.B. JPEG vs. Bitmap. Leitungscodierung definiert die Art und Weise, in der Bits über eine Leitung übertragen werden.*
- Sowohl Quell- als auch Leitungscodierung bestimmen maßgeblich die Übertragungsdauer einer bestimmten Nachricht. Ein Text im Unicode-Zeichensatz ist z.B. doppelt so lange wie derselbe Text im ASCII-Zeichensatz. Ein Bild im Bitmap-Format ist größer als dasselbe Bild im JPEG-Format. Dementsprechend dauert dann die Übertragung der Daten auch länger. Ebenso wird diese Dauer bei einer festen Baudrate jedoch auch von der Leitungscodierung beeinflusst. Wird dieselbe Anzahl Bits z.B. einmal mit NRZ-Codierung und einmal mit 4B3T-Codierung übertragen, so dauert die Übertragung bei Verwendung von NRZ länger.*

### Aufgabe 2: Synchrone Datenübertragung (2 Punkte)

Ein ASCII-Text (8Bit pro Zeichen) wird mittels synchroner Datenübertragung von einem Rechner zum anderen transferiert. Auf Empfängerseite erhalten Sie die unten angegebene Bitsequenz. Dekodieren Sie die erhaltene Nachricht als ASCII-Text! Beachten Sie dabei, dass die Bitsequenz, die das Anwendungsprogramm übertragen wollte, zur korrekten Erkennung von Rahmengenrenzen verändert werden musste!

```
011111100101011101100001011100110011111010101011111001010111011011110011111  
0101111110
```

*Lösungsvorschlag:*

*Wir entfernen das „01111110“-Flag am Anfang und am Ende der Nachricht und ebenso alle Nullen, die auf eine Sequenz von 5 Einsen folgen:*

```
0111111001010111011000010111001100111110101011111001010111011011110011111  
0101111110
```

*Daraus ergibt sich:*

```
0101011101100001011100110011111101011111010101110110111100111111
```

*Gruppieren nach 8-Bit Gruppen:*

```
01010111 01100001 01110011 00111111 01011111 01010111 01101111 00111111
```

*In hexadezimaler Schreibweise:*

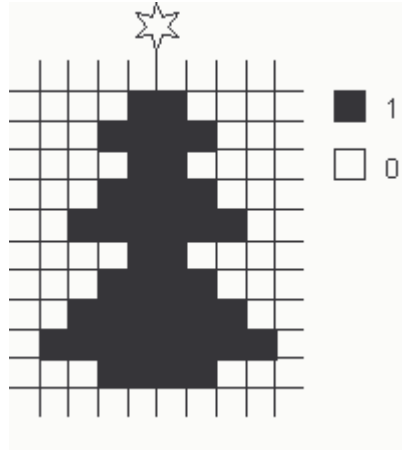
```
57      61      73      3F      5F      57      6F      3F
```

*Dies ist die ASCII-Codierung für folgenden Text: „Was?\_Wo?“*

### Aufgabe 3: Leitungscodierung (6 Punkte, 2+2+2)

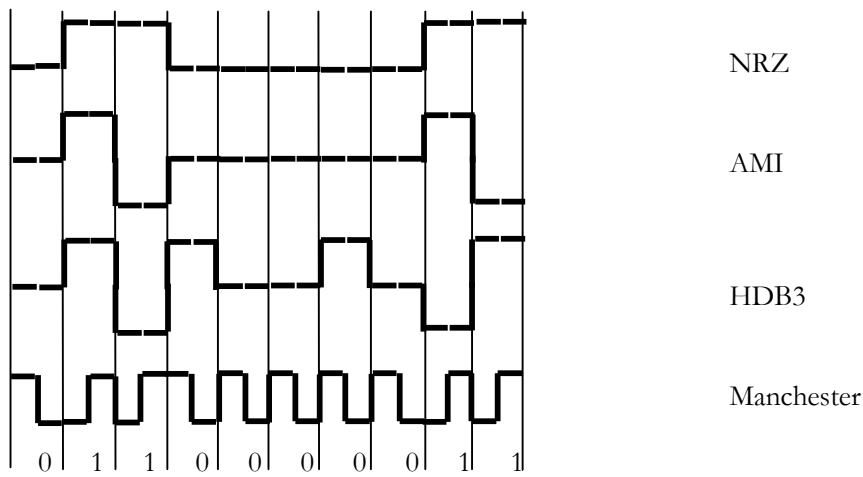
- Codieren Sie die binäre Signalfolge „011000011“ gemäß NRZ, AMI, HDB3 und Manchester (fertigen Sie eine Zeichnung an)!

- b) Nennen Sie Vorteile der einzelnen Codierungen!
- c) Codieren Sie das folgende 8x10 Bitmuster nach 4B3T (Zeichnung). Hinweis: Beginnen Sie mit der Codierung in der obersten Zeile, d.h. das erste zu codierende Byte hat die Form: 0001 1000!



Lösungsvorschlag:

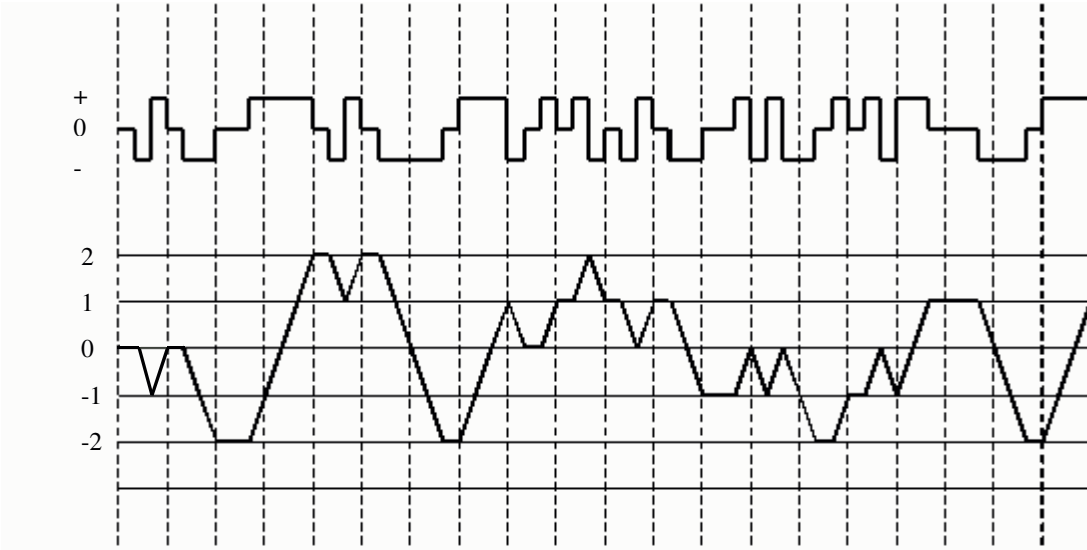
a)



- b) NRZ: einfache, intuitive Realisierbarkeit  
 AMI: Gleichstromanteil ausgeglichen  
 HDB3: Gleichstromanteil ausgeglichen, Takterkennung immer möglich (durch Codeverletzung bei langen 0-Sequenzen)  
 Manchester: Gleichstromanteil ausgeglichen, Takterkennung immer möglich

c)

0001 1000 0011 1100 0001 1000 0011 1100 0111 1110 0001 1000 0011 1100 0111 1110 1111 1111 0011 1100  
 0-+ 0-- 00+ +++ 0-+ 0-- --0 +++ -0+ 0+- 0-+ 0-- 00+ +- -0+ 0+- ++0 00- --0 +++



**Codierungstabelle:**

0001	0010	0100	0111	1011	1110
0 - +	+ - 0	- + 0	- 0 +	+ 0 -	0 + -

	alte Summe < 0	alte Summe >= 0
0000	+ 0 +	0 - 0
0011	0 0 +	- - 0
0101	0 + +	- 0 0
0110	- + +	- - +
1000	+ 0 0	0 - -
1001	+ - +	- - -
1010	+ + -	+ - -
1100	+ + +	- + -
1101	0 + 0	- 0 -
1111	+ + 0	0 0 -

*Hinweis: Wenn (wie im Skript) in der rechten Spalte das Gleichheitszeichen weggelassen wird, dann muss man natürlich dafür sorgen, dass die Summe der Impulse (die untere Linie in der Zeichnung) niemals den Wert 0 erreicht. Dies ist z.B. möglich, indem man die Summe mit einem Startwert von 0.5 initialisiert (wie in der Zeichnung im Skript). Diese Lösung akzeptieren wir natürlich ebenfalls.*