

DT1 / DT2

Manuskript Digitaltechnik 1 + 2

Vorlesungen WS 2003

Prof. Dr.-Ing. Gunter Biethan

1 Einführung und Übersicht

- 1.1 Das Prinzip digitaler Arbeitsweise
- 1.2 Gebräuchliche Zahlensysteme
 - 1.2.1 Das Dualsystem
 - 1.2.2 Das Oktalsystem
 - 1.2.3 Das Hexadezimalsystem
- 1.3 Codes zur Zeichendarstellung
 - 1.3.1 Zifferncodes
 - 1.3.2 Zeichencodes
- 1.4 Die ASCII-Zeichendarstellung
 - 1.4.1 Erweiterung von ASCII im PC-Bereich
 - 1.4.2 Der Unicode (ISO 10646)
- 1.5 Parität bei Ziffern- und Zeichencodes
 - 1.5.1 Schaltungen für Paritätsbildung und –Prüfung
- 1.6 Lerntest/Übungsfragen

2 Beschreibungsverfahren für Digitalschaltungen

- 2.1 Boole'sche Algebra
 - 2.1.1 Logische und technische Schaltungen
 - 2.1.2 Übersicht: Technische Bauelemente zur Realisierung logischer Schaltungen
 - 2.1.3 Binäre Spannungspegel
 - 2.1.4 Positive und negative Logik
- 2.2 Grundvereinbarungen der Schaltalgebra
- 2.3 Einführung in die Funktion digitalelektronischer Schaltungen
 - 2.3.1 Dioden und Transistoren als elektronische Schalter
 - 2.3.2 Einfache Grundschaltungen mit Dioden
 - 2.3.2.1 Die UND-Schaltung (pos. Logik)
 - 2.3.2.2 Die ODER - Schaltung (positive Logik)
 - 2.3.2.3 Mehrstufige (kaskadierte) Dioden-Gatter
 - 2.3.3 Bipolare Transistoren als elektronische Schalter
 - 2.3.3.1 Die NICHT-Schaltung (Negation)
 - 2.3.4 Unipolare Transistoren (FET) als elektronische Schalter
- 2.4 Beschreibung logischer und technischer Schaltungen
 - 2.4.1 Die Funktionstabelle (Wahrheitstabelle)
 - 2.4.2 Der Logikplan
 - 2.4.3 Der Kontaktplan
 - 2.4.4 Die Schaltgleichung
 - 2.4.5 Das Zustandsdiagramm
 - 2.4.6 Hardwarebeschreibungssprachen (HDL)
- 2.5 Anwendungsbeispiel Schaltnetz
- 2.6 Funktionen von einer und zwei Binärvariablen
 - 2.6.1 Funktionen von einer Variablen
 - 2.6.2 Technische Anwendungen: Schaltverstärker, Inverter
 - 2.6.3 Funktionen von zwei Variablen
 - 2.6.3.1 Die Grundfunktionen UND, ODER, NICHT
 - 2.6.3.2 Die Funktionen NAND, NOR
 - 2.6.3.3 Die Funktionen Äquivalenz, Antivalenz
 - 2.6.3.4 Beschaltung freier Eingänge bei UND-/NAND-Gattern
 - 2.6.3.5 Beschaltung freier Eingänge bei ODER-/NOR-Gattern
 - 2.6.3.6 Aufgaben von pull-up und pull-down- Widerständen
- 2.7 Die Normalformen einer Schaltgleichung
 - 2.7.1 Die kanonische disjunktive Normalform (KDNF)
 - 2.7.2 Die kanonische konjunktive Normalform (KKNF)
- 2.8 Rechenregeln der Schaltalgebra

- 2.8.1 Rechenregeln mit Konstanten
- 2.8.2 Rechenregeln mit einer Konstanten und einer Variablen
- 2.8.3 Berücksichtigung der mehrfachen Negation
- 2.8.4 Rechenregeln für mehrere Variablen
 - 2.8.4.1 Das kommutative Gesetz
 - 2.8.4.2 Das assoziative Gesetz
 - 2.8.4.3 Das distributive Gesetz
 - 2.8.4.4 Das Entwicklungstheorem
 - 2.8.4.5 Das De MORGAN-Theorem
 - 2.8.4.6 Das verallgemeinerte Dualitätsgesetz
- 2.9 Analyse und Synthese logischer Schaltungen
- 2.10 Lerntest/ Wissensfragen

3 Vereinfachungsverfahren für Funktionen

- 3.1 Allgemeine Vereinfachungsregeln der Schaltalgebra
 - 3.1.1 Das Absorbtionsgesetz und weitere Regeln
 - 3.1.2 Weitere Vereinfachungsregeln
- 3.2 Grafisches Verfahren nach Karnaugh und Veitch
 - 3.2.1 Aufbau der KV - Diagramme
 - 3.2.2 Benachbarte Felder im KV - Diagramm
 - 3.2.3 Minimieren von Schaltgleichungen in kanonischer disjunktiver Normalform (KDNF)
 - 3.2.4 Minimierungsvorgang bei Schaltgleichungen in KDNF
 - 3.2.5 Berücksichtigung von "don't-care-Bedingungen"
 - 3.2.6 Behandlung und Minimierung von Schaltgleichungen in kanonischer konjunktiver Normalform (KKNF)
- 3.3 Tabellenverfahren Verfahren nach Quine und McCluskey
 - 3.3.1 Die Methode von Quine
 - 3.3.2 Bildung der Minimalform
 - 3.3.3 Das Überdeckungsdiagramm
 - 3.3.4 Verallgemeinertes Verfahren nach Quine-Mc Cluskey
- 3.4 Rechnergestützte Minimierung
- 3.5 Lerntest/ Wissensfragen

4 Wichtige logische Grundschaltungen

- 4.1 Gatterschaltungen
- 4.2 Codierschaltungen
- 4.3 Multiplexer
- 4.4 Vergleicher (Komparatoren)
 - 4.5.1 Arithmetisch-logische Schaltungen
 - 4.5.2 Einfache Arithmetikschaltungen: Halb- und Volladdierer
 - 4.5.2 Mehrstufige Additionsschaltungen/ALU
- 4.6 Grundlagen der Schaltwerke
 - 4.6.1 Die Basis – Flipflops
 - 4.6.1.1 Das NOR-Basis-Flipflop
 - 4.6.1.2 Das NAND-Basis-Flipflop
 - 4.6.1.3 Anwendungen von NOR- und NAND- Basis-Flipflops
 - 4.6.2 Monostabile Kippstufen
 - 4.6.3 Astabile Kippstufen (Multivibratoren)
 - 4.6.4 Funktion und Aufbau technischer Flipflops
 - 4.6.4.1 Merkmale technischer Flipflops
 - 4.6.4.2 Standardformen ungetakteter Flipflops (Latches)
 - 4.6.4.3 Das ungetaktete JK-Flipflop
 - 4.6.4.4 Das T-Flipflop
 - 4.6.4.5 Das ungetaktete D-Flipflop

- 4.7 Taktgesteuerte Flipflops
- 4.8 Beispiele für technische Flipflops
 - 4.8.2 Ausgangsverzögerte Flipflops (Master-Slave-Flipflops)
 - 4.8.3 Übersicht: Integrierte Flipflop-Schaltungen
- 4.9 Arbeits- und Funktionsweise digitaler Speicher
 - 4.9.1 Begriffe für Halbleiterspeicher
 - 4.9.1.1 Speicherorganisation
 - 4.9.1.2 Zugriffsarten
 - 4.9.1.3 Speicherkapazität
 - 4.9.1.4 Zugriffszeit
 - 4.9.1.5 Halbleitertechnologie
 - 4.9.2 Symbole für Halbleiterspeicher
- 4.10 Lerntest/ Wissensfragen

5 Programmierbare Logikelemente (PLD)

- 5.1 Einführung
- 5.2 Grundstruktur von PALs
- 5.3 Die Systematik der programmierbaren Logikbausteine
- 5.4 Arbeiten mit PAL- und GAL-Bausteinen
- 5.5 PLD-Programmierung mit CUPL
- 5.6 Lerntest/ Wissensfragen

6 Die Technik logischer Grundhaltungen

- 6.1 Realisierung von Digitalschaltungen
- 6.2 Integrierte Digitalschaltungen (Übersicht)
- 6.3 Die DTL - Schaltkreistechnik
- 6.4 Die TTL - Schaltkreistechnik
 - 6.4.1 Spezielle Ausgangsschaltungen
 - 6.4.2 Schaltungsvarianten der TTL - Technik
- 6.5 Die CMOS - Schaltkreistechnik
 - 6.5.1 CMOS-Gatterschaltungen
 - 6.5.2 CMOS-Transmissionsgatter
 - 6.5.3 Praktische Probleme der CMOS-Technik
- 6.6 Die BICMOS-Technik
- 6.7 Die ECL-Technik
- 6.8 Vergleich verschiedener Schaltkreisfamilien
- 6.9 Lerntest/Wissensfragen

7..10 Digitaltechnik II (Teil 2)

7 Analyse und Synthese von Schaltwerken

- 7.1 Digitalzähler
 - 7.1.1 Asynchrone Zähler
 - 7.1.2 Synchronzähler
 - 7.1.3 Synthese von synchronen Zähl-schaltungen
- 7.2 Schieberegisterschaltungen
 - 7.2.1 Der Ringzähler
 - 7.2.2 Der Johnsonzähler
 - 7.2.3 Zufallszahlengenerator
- 7.3 Nicht autonome Schaltwerke
 - 7.3.1 Der Mealy-Automat
 - 7.3.2 Der Moore-Automat
 - 7.3.3 Synthese-/Analysemethoden für Automaten

7.4 Lerntest/Wissensfragen

8 Digital-Analog-Umsetzer (DAU)

- 8.1 Statische und dynamische Parameter von DAU
- 8.2 DAU-Grundsaltungen
 - 8.2.1 DAU mit binär gewichteten Widerständen
 - 8.2.2 DAU mit Leiternetzwerk
 - 8.2.3 Eigenschaften belasteter Spannungsteiler
 - 8.2.4 DAU mit dual codiertem Leiternetzwerk
 - 8.2.5 DAU mit dekadischer Abstufung (BCD-Umsetzer)
- 8.3 Spezielle Formen von Digital-Analog-Umsetzern
- 8.4 Technische Realisierung von Digital-Analog-Umsetzern
 - 8.4.1 DAU in CMOS-Technik
 - 8.4.2 DAU in bipolarer Technik
- 8.5 Wichtige technische Anwendungen von DAU
- 8.6 Lerntest/Wissensfragen

9 Analog-Digital-Umsetzer (ADU)

- 9.1 Kenngrößen des ADU
- 9.2 Die Grundprinzipien
- 9.3 ADU nach dem Parallelverfahren (Flash Converter)
 - 9.3.1 Kaskadenumsetzer
- 9.4 ADU nach dem Wägeverfahren (sukzessive Approximation)
- 9.5 Zählende Verfahren (Integrationsumsetzer)
 - 9.5.1 Einzelrampen-Umsetzer (Single Slope Verfahren)
 - 9.5.2 Zwei-Rampen-Umsetzer (Dual Slope-Verfahren)
- 9.6 Der Spannungs-Frequenz-Umsetzer
- 9.7 Technische Bauformen von ADU
 - 9.7.1 Schaltungsintegration von ADU
 - 9.7.2 Spezialschaltungen für monolithische ADU
 - 9.7.2.1 Das Ladungsverteilungsverfahren
 - 9.7.2.2 Das Sigma-Delta-Verfahren
- 9.8 Dynamische Spezifikation von ADU
 - 9.8.1 Signal-Rausch-Abstand (SNR), Rauschspannung
 - 9.8.2 Effektive Auflösung (ENOB)
 - 9.8.3 Klirrfaktor (THD)
- 9.9 Lerntest/Wissensfragen

10 EMV-Verhalten von integrierten Digitalschaltungen (noch nicht enthalten)

11 Übungs- und Klausuraufgaben (DT1 + DT2)

12 Literatur

Lernziele: Unterschiede Analog- und Digitaltechnik
Vorzüge digitaler Systeme
Zahlensysteme für die Digitaltechnik
Ziffern- und Zeichencodes
Verfahren der Codeprüfung

1. Einführung und Übersicht

Digital arbeitende Geräte sind seit mehr als 300 Jahren bekannt. So konstruierte **Schickard** bereits 1623 für den Astronom J. Kepler eine Rechenuhr für Addition und Subtraktion. Blaise **Pascal** baute 1641 eine Addiermaschine mit 6 Dezimalstellen. Der analoge Rechenschieber hingegen wurde erst 1650 durch **Patridge** erfunden.

Digitaltechnik ist also keinesfalls an elektronisch arbeitende Systeme gebunden, sondern kann grundsätzlich alle anwendbaren physikalischen Effekte der Mechanik, Pneumatik, Hydraulik ausnutzen.

Durch die Entwicklung zuverlässiger elektromechanischer Bauelemente (Relais) und später elektronischer Halbleiterschaltkreise hat die Digitaltechnik einen fantastischen Aufschwung genommen.

Nachfolgend einige Beispiele:

1941 erster Digitalrechner (programmgesteuert), K.Zuse aus ca. 2600 Postrelais

1959 Schaltungsintegration J.Kilby/TI & J.A.Hoerni/Fairchild

1971 Mikroprozessor Fa. Intel

1975 Komplexe Signalprozessoren
Verstärkter Trend zur Digitalisierung im Bereich der Nachrichtentechnik.

Anwendungen der Digitaltechnik

Die Digitaltechnik hat die früher dominierende Analogtechnik weitgehend verdrängt. Nachfolgend einige typische Einsatzbereiche:

- Computertechnik
 - Universalrechner
 - Minicomputer (Workstation)
 - Personal Computer (Mikrocomputer)
- Nachrichten-/Kommunikationstechnik
 - Kommunikationsnetze (ISDN), Internet)
 - Vermittlungseinrichtungen (EWSD,S12)
 - Sprach- und Bildübertragung
- Automatisierungstechnik
 - Prozeßrechner
 - Speicherprogrammierbare Steuerungen 'SPS' (z.B. Siemens Simatic S5, S7)
 - CNC-Technik
- Fahrzeugtechnik
 - Führungs- und Leitsysteme
 - Motorüberwachung

Auswirkungen der Digitaltechnik

Durch den Einsatz der Mikroelektronik gelang es, die Kosten für digitale Systeme in starkem Maße zu senken. Hierdurch konnte in vielen neuen Anwendungsgebieten ein wirtschaftlicher Einsatz dieser Systeme begründet werden. Dies führte in den meisten Bereichen der Arbeitswelt zu erheblichen Umwälzungen. Zahlreiche Berufsbilder haben sich völlig gewandelt oder sind verschwunden. Hier einige Beispiele:

- Kommunikationstechnik(Rechner + digitale Nachrichtensysteme)
- Produktionstechnik (automatische Fertigung, Roboter,CIM)
- Entwicklung und Konstruktion
- C-Techniken (CAE,CAD/CAM,CAQ)
- Expertensysteme ("Künstliche Intelligenz")

1.1 Das Prinzip digitaler Arbeitsweise

In der Technik gibt es grundsätzlich zwei verschiedene Darstellungsarten:

- Analoge Darstellung/Verarbeitung
Ursprung "ana logon" (griech.) = im richtigen Verhältnis stehend. Hierbei wird eine physikalische Größe durch eine entsprechende elektrische Größe (z.B. Spannung, Strom) nachgebildet.

Beispiele: Analogrechner
analoge Motorenregelung
Tachometer
Rechenschieber

- Digitale Darstellung/Verarbeitung
Ursprung "digitus" (lat.) = Finger. In diesem Fall werden alle Größen durch diskrete Zahlenwerte beschrieben.

Beispiele: Taschenrechner
Abacus
CNC - Steuerung
Digitaluhr

Unterschiede Analog-/Digitaltechnik

Bei der Analogtechnik werden physikalische Größen mit zeitlicher Stetigkeit verarbeitet. Da physikalische Größen immer kontinuierlich sind, ist eine analoge Größe zu jeder Zeit definiert, ihr Wert könnte theoretisch beliebig genau abgelesen werden.

Bei der Digitaltechnik hingegen liegen alle Werte numerisch, d.h. ziffernmäßig vor und werden mit Hilfe logischer bzw. arithmetischer Operationen verarbeitet. Es bestehen folgende Zuordnungen:

kontinuierlich \leftrightarrow Analogtechnik
diskret \leftrightarrow Digitaltechnik

Die wichtigsten Vor- und Nachteile beider Techniken sind nachfolgend dokumentiert.

Vor- und Nachteile der Analogtechnik**Vorteile**

einfach, anschaulich
geringer technischer Aufwand

Nachteile

begrenzte Genauigkeit
(kaum besser als 1 %.)

Vor- und Nachteile der Digitaltechnik**Vorteile**

beliebige Genauigkeit
hohe Störsicherheit
keine Ergebnisverschlechterung

Nachteile

relativ hoher Aufwand bei der
Verarbeitung von Analogwerten

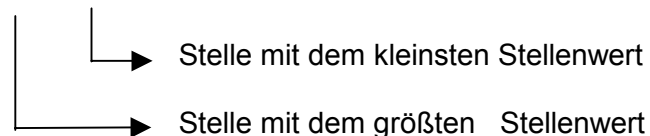
1.2 Gebräuchliche Zahlensysteme

Die heute allgemein verwendeten Zahlensysteme sind Stellenwertsysteme. Dies bedeutet: Jeder Stelle ist ein bestimmter Wert zugeordnet. Man bezeichnet ihn als Stellenwert.

Beispiel:

Stellen

| | | |
5 0 4 9



Bei Dezimalzahlen werden die Stellen mit Einerstelle, Zehnerstelle, Hunderterstelle, Tausenderstelle usw. bezeichnet. Die Stellenwerte sind also 1, 10, 100, 1000 usw. Der nächstgrößere Stellenwert wird gefunden, indem der vorhergehende Stellenwert mit 10 multipliziert wird. Zehn ist damit die Basis 'B' des Dezimalstellensystems. Der Wert, den eine Ziffer darstellt, errechnet sich als Produkt aus dem Nennwert der Ziffer und dem Stellenwert.

Beispiel:

$$\text{Zahl: } 5\ 0\ 4\ 9 = 5 \cdot 1000 + 0 \cdot 100 + 4 \cdot 10 + 9 \cdot 1$$

$$\begin{array}{cccc} \downarrow & \downarrow & \downarrow & \downarrow \\ 10^3 & 10^2 & 10^1 & 10^0 \end{array}$$

Die Summe der Produkte aus Nennwert der Ziffer und Stellenwert ergibt einen Ausdruck, der als Potenzreihe mit der Basis $B = 10$ entwickelt werden kann.

In der Digitaltechnik ist das Dezimalsystem wegen seiner 10 verschiedenen Ziffern und damit der zehn verschiedenen (elektrischen) Signalpegel wenig zweckmäßig. Man verwendet daher allgemein das Dualsystem (Ziffern 0 und 1) sowie die hiervon abgeleiteten Zahlensysteme 'Oktal' sowie 'Hexadezimal (Sedezimal)'. Unabhängig von der Art des verwendeten Zahlensystems kann jeder Zahlenwert Z als Potenzreihe gemäß Gl. 1.1 dargestellt werden.

$$Z = Z_i \cdot B^i + Z_{i-1} \cdot B^{i-1} + \dots + Z_1 \cdot B^1 + Z_0 \cdot B^0 \quad (1.1)$$

Hierbei ist B die Basis des Zahlensystems.

Für alle derartigen Zahlensysteme gilt:

Anzahl der Ziffensymbole: B, dh. von 0, 1 ... (B-1)
 Ziffer mit dem höchsten Nennwert: B-1

1.2.1 Das Dualsystem

Das duale Zahlensystem hat sich in der Digitaltechnik durchgesetzt, da auch zweiwertige (=binäre) Logik sowie entsprechende Schaltungen benutzt werden. Es gelten folgende Festlegungen.

Basis des Zahlensystems : B = 2
 Ziffersymbole : 0, 1

Beispiel:

$$\begin{array}{cccccccc} \text{Dualzahl: } 11101011 & = & 1 \cdot 2^7 & + & 1 \cdot 2^6 & + & 1 \cdot 2^5 & + & 0 \cdot 2^4 & + & 1 \cdot 2^3 & + & 0 \cdot 2^2 & + & 1 \cdot 2^1 & + & 1 \cdot 2^0 \\ & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ & & 128 & & 64 & & 32 & & 16 & & 8 & & 4 & & 2 & & 1 \end{array}$$

Dualzahlen werden zur besseren Unterscheidung von anderen Systemen üblicherweise mit dem Index '2' versehen. In der Informationstechnik, beispielsweise bei der Programmierung in Assembler, benutzt man für duale Objekte häufig den Buchstaben 'B' (Binary).

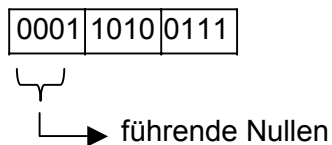
Beispiele: 1100_2 , 1111b, s_block EQU 10101111b

Der Begriff der Tetrade

Faßt man, beginnend bei der Stelle mit dem kleinsten Stellenwert, jeweils vier Dualstellen zusammen, so erhält man eine 'Tetrade' (griech. tetra = vier), wobei ggf. die letzte Tetrade durch führende Nullen zu ergänzen ist.

Beispiel:

Dualzahl: 1 1010 0111



Durch diese Methode können jeweils vier duale Stellen mit einer hexadezimalen Ziffer codiert dargestellt werden.

1.2.2 Das Oktalsystem

Das oktale Zahlensystem hat inzwischen stark an Bedeutung verloren. Bei früheren Rechnersystemen wurde zur übersichtlicheren Dokumentation maschineninterner Zustände (z.B. Adressen/Speicher- und Registerinhalte) die oktale Darstellung verwendet. Das Oktalsystem wird heute noch in starkem Maße in der Kommunikationstechnik benutzt. Beim Oktalsystem gelten folgende Festlegungen.

Basis des Zahlensystems : B = 8
 Ziffersymbole : 0, 1, 2, 3, 4, 5, 6, 7

Beispiel:

$$\text{Oktalzahl: } 176 = 1 \cdot 8^2 + 7 \cdot 8^1 + 6 \cdot 8^0$$

$$\begin{array}{ccc} \downarrow & \downarrow & \downarrow \\ 64 & 8 & 1 \end{array}$$

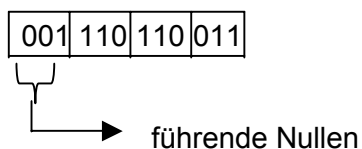
Zur Unterscheidung von anderen Zahlen wird bei Oktalzahlen der Index '8' verwendet, in der Informationstechnik gelangt häufig der nachgestellte Buchstabe 'O' bzw. 'Q' als Suffix zur Anwendung.

Beispiele: 754_8 , $164O$, $5371Q$

Wegen $2^3 = 8$ ist das oktale Zahlensystem mit dem Dualsystem sehr eng verwandt. Eine Dualzahl kann daher sehr einfach in das Oktalsystem überführt werden, indem - beginnend bei der niederwertigsten Stelle jeweils drei Stellen zusammengefaßt werden. Ggf. ist die Dualzahl wieder mit führenden Nullen zu ergänzen.

Beispiel:

Dualzahl : 1 110 110 011



$$\text{Oktalzahl: } 1663 = 1 \cdot 8^3 + 6 \cdot 8^2 + 6 \cdot 8^1 + 3 \cdot 8^0$$

1.2.3 Das Hexadezimalsystem

Das Hexadezimalsystem (nach DIN Sedezimalsystem) wird zur Codierung von Dualzahlen überwiegend benutzt. Es gelten hierbei folgende Festlegungen.

Basis des Zahlensystems: $B = 16$

Ziffersymbole : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Beispiel:

$$\text{Hexadezimalzahl: } ABF9 = 10 \cdot 16^3 + 11 \cdot 16^2 + 15 \cdot 16^1 + 9 \cdot 16^0$$

$$\begin{array}{cccc} \downarrow & \downarrow & \downarrow & \downarrow \\ 4096 & 256 & 16 & 1 \end{array}$$

Wie o.a. können Dualzahlen und hexadezimale Zahlen über die Tetraden besonders einfach ineinander überführt werden. Da in der Informationstechnik jeweils 8 binäre Stellen (Bits) zu einem Byte zusammengefaßt werden, kommt dem Hexadezimalsystem höchste Bedeutung zu. Ein Byte läßt sich also durch zwei hexadezimale Ziffersymbole darstellen. Ein Wort (word) der Länge 16 Bit entspricht also einer 4-stelligen Hexadezimalzahl. Zur Kennzeichnung hexadezimaler Zahlen wird die Zahl '16' als Index benutzt. Zur Bezeichnung von Objekten des Typs hexadezimal verwendet man häufig ein nachgestelltes 'h' (z.B. beim Makro-Assembler MASM) oder aber ein vorangestelltes '\$'-Zeichen (z.B. Programmiersprache Pascal). Bei den Programmiersprachen C bzw. C++ wird ein vorangestelltes '0x' benutzt..

Beispiele: $AFF0_{16}$, BFh , $\$AB$, $0xAFFE$

In Tabelle 1.1 sind die genannten Zahlensysteme nochmals gegenübergestellt, wobei die Dualzahlen zu Tetraden ergänzt wurden.

Tabelle 1.1 Gegenüberstellung von gebräuchlichen Zahlensystemen

Zustand Nr.	Dezimalzahl	Dualzahl	Oktalzahl	Hexadezimalzahl	Bemerkungen
1	0	0000	0	0	
2	1	0001	1	1	
3	2	0010	2	2	
4	3	0011	3	3	
5	4	0100	4	4	
6	5	0101	5	5	
7	6	0110	6	6	
8	7	0111	7	7	
9	8	1000	10	8	
10	9	1001	11	9	
11	10	1010	12	A	"Pseudotetraden" bei codierter Darstellung (8-4-2-1 - Code) von Dezimalziffern
12	11	1011	13	B	
13	12	1100	14	C	
14	13	1101	15	D	
15	14	1110	16	E	
16	15	1111	17	F	

Bei der Anwendung der o.a. Zahlensysteme ist zu beachten, daß Null stets als Ziffernsymbol berücksichtigt werden muß. Eine Zählschaltung beispielsweise, die bei Null anfängt, zählt den Zustand Nr. 1. Die Zustände Nr. 11 bis Nr. 16 werden bei codierter Darstellung von Dezimalzahlen im bekanntesten BCD - Code, dem 8-4-2-1 - Code als "Pseudotetraden" (griech. "pseudum" = täuschen) bezeichnet, da diese im Normalfall niemals auftreten können.

1.3 Codes zur Zeichendarstellung

Sollen Ziffern oder Buchstaben im Binärsystem dargestellt werden, so muß jeder Ziffer bzw. jedem Buchstaben eine bestimmte Kombination mehrerer Bits zugeordnet werden. Aus praktischen Gründen wird die Zuordnung meist umkehrbar eindeutig gewählt (Bild 1.1). Beim Fernschreibcode zum Beispiel ist diese Bedingung nicht gegeben.

Der Begriff eines Codes ist laut DIN 44300 folgendermaßen definiert: Ein Code ist eine Vorschrift für die eindeutige Zuordnung der Zeichen eines Zeichenvorrats zu denjenigen eines anderen Zeichenvorrats.

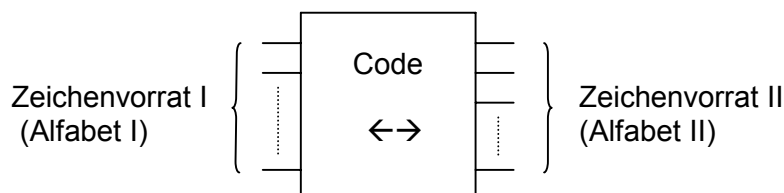


Bild 1.1: Funktion eines umkehrbar eindeutigen Codes

Zu unterscheiden sind die Begriffe Codieren und Decodieren mit nachfolgender Bedeutung.

Codieren Verschlüsseln, d.h. Umwandeln der Zeichen des Alphabets I in entsprechende Zeichen des Alphabets II

Decodieren: Entschlüsseln, d.h. Umkehrung des obigen Vorgangs.

Als Code wird in der Praxis auch das neue Alphabet II bezeichnet (z.B. BCD - Code, Fernschreibcode, ASCII-Code usw.).

Die Anzahl der Bits (=Stellenzahl) eines binären Codewortes richtet sich nach der Größe des Zeichenvorrats. Für die Dezimalziffern 0 - 9 genügen 4 Bit.

Allgemein: aus n Bit können 2^n verschiedene Kombinationen aus 0 und 1 gebildet werden, d.h. also 2^n verschiedene Codeworte.

1.3.1 Zifferncodes

Die Zifferncodes besitzen für die Digitaltechnik allergrößte Bedeutung, da bei kleineren technischen Systemen überwiegend numerische Daten, d.h. Zahlenwerte verarbeitet werden. Bei den dezimalen Zifferncodes unterscheidet man

- Tetradsische Codes, d.h. jedes Codewort besteht aus vier binäre Zeichen (Bits),
Beispiel: Aiken-Code
- Nicht tetradsische Codes, d.h. mehr als vier binäre Zeichen,
Beispiel: 2-aus-5-Code (5 Bit)

Tabelle 1.2 zeigt als Beispiel für einen weit verbreiteten tetradsischen Code den 8421-Code. Die sechs "überflüssigen" Kombinationen werden als Pseudotetraden bezeichnet.

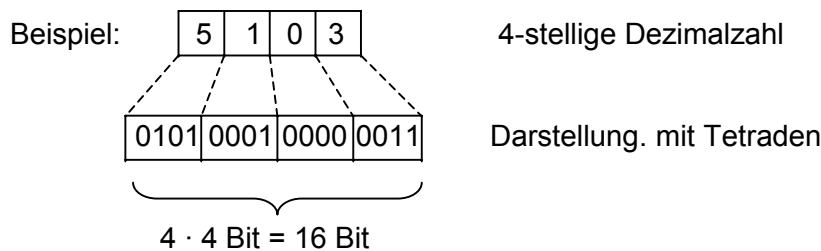


Tabelle 1.2 8421-Code als Beispiel für Tetradencodes

	Alfabet I	Alfabet II
Bedeutung	Numerisches Alfabet	Binär-codierte Darst. (Tetraden)
zehn Dezimalziffern	0	0000
	1	0001
	2	0010
	3	0011
	4	0100
	5	0101
	6	0110
	7	0111
	8	1000
	9	1001
unbenutzte Tetraden	-	1010
	-	1011
	-	1100
	-	1101
	-	1110
	-	1111

Für jede Dezimalstelle wird also eine Tetrade (= 4 Bit) verwendet. Die Codierung einer Dezimalziffer mit mindestens vier Bits ergibt sich aus der Tatsache, daß die höchstwertige Dezimalziffer - Neun - zur binären Darstellung mindestens vier Stellen (1001) erfordert. Dezimalzahlen können nun im Rechner rein binär (als Dualzahlen) oder als binär - codierte Dezimalzahlen verschlüsselt sein (BCD-Zifferncodes).

Häufig wird in der Digitaltechnik der 8421 - Code als "der" BCD- Code bezeichnet. Es ist jedoch nur einer unter zahlreichen anderen Codes, die in der Praxis eingesetzt werden und jeweils spezielle Vor- und Nachteile aufweisen.

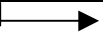
Für die Beurteilung von Zifferncodes sind folgende Eigenschaften maßgeblich:

- Bewertbarkeit der Stellen, d.h. jeder Stelle des binären Codewortes ist ein Stellenwert zugeordnet
- Einschrittigkeit, d.h. die Folge der Codeworte ändert sich jeweils nur in einer Bitstelle.
- Komplementierbarkeit, d.h. eine bitmäßige Komplementierung führt auch dezimal zum entsprechenden Komplement.
- Prüfbarkeit, d.h. die Codeworte können im Hinblick auf fehlerhafte Worte geprüft werden.
- Korrigierbarkeit, d.h. Fehler in Codeworten werden erkannt und korrigiert.

Tabelle 1.3 enthält neben dem 8421-Code noch drei weitere tetradische Zifferncodes, nämlich

- Gray – Code
- Aiken – Code
- Dreiezzess - Code

Tabelle 1.3 Häufig verwendete Tetradencodes

Ziffer Wertigkeit	8421- Code	Gray- Code	Aiken- Code	Dreiezzess- Code
	8421	-	2421	-
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0011	0010	0101
3	0011	0010	0011	0110
4	0100	0110	0100	0111
5	0101	0111	1011	1000
6	0110	0101	1100	1001
7	0111	0100	1101	1010
8	1000	1100	1110	1011
9	1001	1101	1111	1100

Das wichtigste Merkmal des Gray - Codes ist die Einschrittigkeit. Dieser Code oder vergleichbare einschrittige Codes werden bei der numerischen Erfassung von codierten Längen und Winkeln (z.B. bei Werkzeugmaschinen) mit Hilfe von Codelinealen bzw. Codescheiben verwendet. Als Hauptvorteil gilt die Vermeidung starker Lesefehler im Fall einer Dejustage des Erfassungssystems. Beim Übergang von einer Ziffer zur nächsten kann sich jeweils nur eine Bitstelle ändern.

Aiken - Code und Dreiezzess-Code eignen sich für die numerische Verarbeitung in Rechenschaltungen (ALU \triangleq arithmetic logic unit), beide besitzen das Merkmal Komplementierbarkeit. Durch bitmäßige Invertierung der Codes entsteht das Neuner-Komplement. Beim Aiken-Code besitzt zusätzlich jede Stelle eine Wertigkeit.

Nicht tetradische Zifferncodes

Ein Nachteil der tetradischen Codes besteht darin, daß die Prüfmöglichkeiten im Falle eines Übermittlungsfehlers außerordentlich beschränkt sind. Einzel- und Mehrbitfehler können nur dann erkannt werden, wenn diese eine Pseudotetrade hervorrufen. Bei Verwendung von mehr als vier Bitstellen zur Codierung einer Dezimalziffer wird die Redundanz der Codes vergrößert. So besitzt beispielsweise ein 5-Bit-Code $2^5 = 32$ verschiedene Codewörter, von denen nur 10 benutzt werden. Von besonderer Bedeutung sind die gleichgewichtigen oder m-aus-n-Codes. Alle verwendeten Codewörter enthalten gleich viele 1-Bits, sie besitzen gleiche Parität. Die mögliche Zahl N gültiger Codewörter ergibt sich aus nachfolgender Gleichung (Gl. 1.1)

$$N = \binom{n}{m} = \frac{n!}{m!(n-m)!} \quad (1.1)$$

Beispiel: 2-aus-5-Code: $n = 5$, $m = 2$

$$N = \binom{5}{2} = \frac{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{2 \cdot 1 \cdot (3 \cdot 2 \cdot 1)} = \frac{120}{12} = 10$$

Es lassen sich damit genau 10 Ziffern codieren. Das gleiche Ergebnis erhält man für den 1-aus-10-Code, der allerdings die doppelte Stellenzahl aufweist und damit eine weitaus höhere Redundanz besitzt. In Tabelle 1.4 sind einige häufig verwendete Codes aus dieser Gruppe dokumentiert.

Tabelle 1.4 Auswahl gleichgewichtiger Codes (m aus n-Codes)

Ziffer	(2 aus 5)-Code	(1 aus 10)-Code	Biquinär-Code	Quibinär-Code
Wertigkeit	7 4 2 1 0		0543210	8642010
		9876543210		
0	1 1 0 0 0	0000000001	1000001	0000101
1	0 0 0 1 1	0000000010	1000010	0000110
2	0 0 1 0 1	0000000100	1000100	0001001
3	0 0 1 1 0	0000001000	1001000	0001010
4	0 1 0 0 1	0000010000	1010000	0010001
5	0 1 0 1 0	0000100000	0100001	0010010
6	0 1 1 0 0	0001000000	0100010	0100001
7	1 0 0 0 1	0010000000	0100100	0100010
8	1 0 0 1 0	0100000000	0101000	1000001
9	1 0 1 0 0	1000000000	0110000	1000010

1.3.2 Zeichencodes

Die Codierung von Zeichen erfordert mehr als vier Bit. Der internationale Fernschreibcode Nr. 2 des CCITT verwendet 5 Bit, damit sind nur $2^5 = 32$ verschiedene Zeichen darstellbar, deshalb doppelte Belegung der Kombinationen, die durch Umschaltzeichen (Buchstaben/Ziffern & Sonderzeichen) zugeordnet werden. Tabelle 1.5 zeigt die Codierungen des Fernschreibcodes.

Tabelle 1.5 Fernschreibcode Nr.2 nach CCITT

Nr.	Lochkombination 5 4 3 T 2 1	Bitkombination	Buchstaben	Ziffern und Zeichen
1	· o o	00011	A	-
2	o o · o	11001	B	?
3	o o · o	01110	C	:
4	o · o	01001	D	wer da?
5	· o	00001	E	3
6	o o · o	01101	F	(frei)
7	o o · o	11010	G	(frei)
8	o o ·	10100	H	(frei)
9	o · o	00110	I	8
10	o · o o	01011	J	Klingel
11	o o · o o	01111	K	(
12	o · o	10010	L)
13	o o o ·	11100	M	.
14	o o ·	01100	N	
15	o o ·	11000	O	9
16	o o · o	10110	P	0
17	o o · o o	10111	Q	1
18	o · o	01010	R	4
19	o · o	00101	S	'
20	o ·	10000	T	5
21	o · o o	00111	U	7
22	o o o · o	11110	V	=
23	o · o o	10011	W	2
24	o o o · o	11101	X	/
25	o o · o	10101	Y	6
26	o · o	10001	Z	+
27	o ·	01000	Wagenrücklauf	(CR)
28	· o	00010	Zeilenschaltung	(LF)
29	o o o · o o	11111	Umschaltung Buchstaben	
30	o o · o o	11011	Umschaltung Ziff.(Zeichen)	
31	o ·	00100	Zwischenraum	(SP)
32	·	00000	(Nicht verwendet)	

Der 5-Bit-Fernschreibcode besitzt folgende Nachteile:

- Geringer Zeichenvorrat (keine Unterscheidung Groß- und Kleinbuchstaben)
- Wenig Sonderzeichen zur Geräte- bzw. Übertragungssteuerung
- Code nicht umkehrbar wegen Doppelbelegung
- Übertragungsfehler nicht erkennbar.

Aus diesem Grund werden häufig Codes mit Längen von 6, 7 und 8 Bit verwendet.

Beispiel für einen 6-Bit Code: Standard-BCD Universal Code

Prof. Dr. -Ing. G. Biethan Fassung 1.2 vom 01.11.2002

Beim Zeichencode Standard-BCD-Universal Code werden 6 Bits verwendet. Damit ist der mögliche Zeichenvorrat auf $2^6 = 64$ Zeichen beschränkt. Zur Anwendung gelangte dieser Code bei Digitalrechnern der 2. und 3. Generation (z.B. Siemens 3003 IBM 1400) als Übertragungs-Code zwischen Zentraleinheit und Datenstationen.

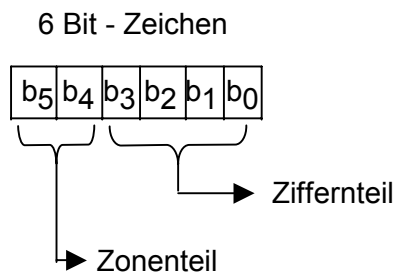


Bild 1.2 Aufbau Codewörter von Standard-BCD-Universal-Code

Bild 1.2 kennzeichnet den Aufbau dieses Zeichencodes, die vollständige Tabelle ist mit Tabelle 1.6 dokumentiert.

Tabelle 1.6 Zeichencodierung nach Standard BCD-Universal Code

(Zonenbits) b_4	0	1	0	1
b_5	0	0	1	1
b_3, b_2, b_1, b_0				
0 0 0 0	SP		-	+
0 0 0 1	1	/	J	A
0 0 1 0	2	S	K	B
0 0 1 1	3	T	L	C
0 1 0 0	4	U	M	D
0 1 0 1	5	V	N	E
0 1 1 0	6	W	O	F
0 1 1 1	7	X	P	G
1 0 0 0	8	Y	Q	H
1 0 0 1	9	Z	R	I
1 0 1 0	0			
1 0 1 1	=	,		.
1 1 0 0	'	(*)
1 1 0 1				
1 1 1 0				
1 1 1 1				

Ein Nachteil des Standard-BCD-Universal-Codes ist der geringe Zeichenvorrat von 64 verschiedenen Zeichen. Deshalb ist keine Unterscheidung zwischen Groß- und Kleinbuchstaben möglich. Für Zwecke der Textverarbeitung wäre daher dieser Code nicht geeignet. Auch fehlen geeignete Steuerzeichen (non printable characters) zur Steuerung der Datenübertragung zur Peripherie (z.B. Modems, Drucker, Bildschirmgeräte, etc.).

Beispiel für einen 8-Bit-Code: EBCDIC (Extended Binary Coded Decimals Interchange Code)

Hierbei handelt es sich um einen 8-Bit- oder Byte-Code, der unter dem Kürzel EBCDIC (Extended Binary Coded Decimals Interchange Code) bekannt ist und bei IBM- Großrechnern Verwendung findet. Wegen $2^8 = 256$ sind entsprechend viele Zeichencodierungen möglich, die allerdings nicht alle zugeordnet sind. Tabelle 1.7 kennzeichnet die Codierung aller Zeichen nach EBCDIC.

Tabelle 1.7 Zeichencodierung nach EBCDIC

		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1						
b ₇																							
b ₆		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1						
b ₅		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1						
b ₄		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1						
Hexa-		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F						
b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀	dez.																						
0 0 0 0 0	0	NUL					SPA	&	-								0						
0 0 0 1 1	1								/	a	j					A	J	1					
0 0 1 0 2	2									b	k	s					B	K	S	2			
0 0 1 1 3	3									c	l	t					C	L	T	3			
0 1 0 0 4	4	PF	RES	BYP	PN								d	m	u					D	M	U	4
0 1 0 1 5	5	HT	NL	LF	RS								e	n	v					E	N	V	5
0 1 1 0 6	6	LC	BS	EOB	UC								f	o	w					F	O	W	6
0 1 1 1 7	7	DEL	IL	PRE	EOT								g	p	x					G	P	X	7
1 0 0 0 8	8									h	q	y					H	Q	Y	8			
1 0 0 1 9	9									i	r	z					I	R	Z	9			
1 0 1 0 A	A					SM	ç	!	^	:													
1 0 1 1 B	B						.	\$,	+													
1 1 0 0 C	C						<	*	%	@													
1 1 0 1 D	D						()	-	'													
1 1 1 0 E	E						+	;	>	=													
1 1 1 1 F	F							¬	?	"													

1.4 Die ASCII-Zeichendarstellung

Beim 7 Bit - Code ASCII (**A**merican **S**tandard **C**ode of **I**nformation **I**nterchange besteht der vereinbarte Zeichenvorrat aus $2^7 = 128$ Zeichen, die sich gemäß Tabelle 1.8 in folgende Hauptgruppen einteilen lassen

- Steuerzeichen (Codewörter 00h ... 1Fh)
- Sonderzeichen und Ziffern (Codewörter 20h ... 3Fh)
- Großbuchstaben oder Versalien (Codewörter 40h ... 5Fh)
- Kleinbuchstaben (Codewörter 60h ... 7Fh)

Ursprüngliche Anwendung fand ASCII als Übertragungs-Code (amerikanischer Fernschreibcode) zwischen Rechner-Zentraleinheit und Datenendgeräten. Im Bereich der Mikro- und Bürocomputer wurde ASCII schon sehr früh eingeführt. Aufgrund der massiven Verbreitung dieser Geräte (z.B. Personal Computer) und der Standardisierung nach CCITT bzw. ISO hat sich ASCII inzwischen weltweit als Standard für die Codierung alphanumerischer Zeichen durchgesetzt.

Tabelle 1.8 ASCII-Zeichendarstellung (7 Bit)

dec	Hex	Char	dec	hex	Char	dec	hex	Char	dec	hex	Char
0	00	(NUL)	32	20	[SP]	64	40	@	96	60	`
1	01	(SOH)	33	21	!	65	41	A	97	61	a
2	02	(STX)	34	22	"	66	42	B	98	62	b
3	03	(ETX)	35	23	#	67	43	C	99	63	c
4	04	(EOT)	36	24	\$	68	44	D	100	64	d
5	05	(ENQ)	37	25	%	69	45	E	101	65	e
6	06	(ACK)	38	26	&	70	46	F	102	66	f
7	07	(BEL)	39	27	'	71	47	G	103	67	g
8	08	(BS)	40	28	(72	48	H	104	68	h
9	09	(HT)	41	29)	73	49	I	105	69	i
10	0A	(LF)	42	2A	*	74	4A	J	106	6A	j
11	0B	(VT)	43	2B	+	75	4B	K	107	6B	k
12	0C	(FF)	44	2C	,	76	4C	L	108	6C	l
13	0D	(CR)	45	2D	-	77	4D	M	109	6D	m
14	0E	(SO)	46	2E	.	78	4E	N	110	6E	n
15	0F	(SI)	47	2F	/	79	4F	O	111	6F	o
16	10	(DLE)	48	30	0	80	50	P	112	70	p
17	11	(DC ₁)	49	31	1	81	51	Q	113	71	q
18	12	(DC ₂)	50	32	2	82	52	R	114	72	r
19	13	(DC ₃)	51	33	3	83	53	S	115	73	s
20	14	(DC ₄)	52	34	4	84	54	T	116	74	t
21	15	(NAK)	53	35	5	85	55	U	117	75	u
22	16	(SYN)	54	36	6	86	56	V	118	76	v
23	17	(ETB)	55	37	7	87	57	W	119	77	w
24	18	(CAN)	56	38	8	88	58	X	120	78	x
25	19	(EM)	57	39	9	89	59	Y	121	79	y
26	1A	(SUB)	58	3A	:	90	5A	Z	122	7A	z
27	1B	(ESC)	59	3B	;	91	5B	[123	7B	{
28	1C	(FS)	60	3C	<	92	5C	\	124	7C	
29	1D	(GS)	61	3D	=	93	5D]	125	7D	}
30	1E	(RS)	62	3E	>	94	5E	^	126	7E	~
31	1F	(US)	63	3F	?	95	5F	_	127	F	(DEL).

Wie Tabelle 1.81 erkennen lässt, sind die Zeichen bei ASCII den Codes nicht willkürlich zugeordnet, sondern folgen einfachen Bildungsgesetzen:

- 00h .. 1Fh : Steuerzeichen für die Datenkommunikation (non printable characters), z.B.
 - 0Ah \triangleq LF (line feed) = Zeilenvorschub, 07h \triangleq BEL (bell) = Signalton,
 - 08h \triangleq BS (backspace) = Rückschritt, 11h \triangleq DC₁ (device control 1)
- 20h .. 2Fh : Sonderzeichen, z.B. 20h \triangleq SP (space) = Leerzeichen
- 30h .. 3Fh : ASCII-Ziffern + Sonderzeichen, z.B. 30h \triangleq '0', ... , 39h \triangleq '9'
- 40h .. 5Fh : Großbuchstaben (Versalien) + Sonderzeichen, z.B. 41h \triangleq 'A', ... , 5Ah \triangleq 'Z'
- 60h .. 7Fh : Kleinbuchstaben + Sonderzeichen, z.B. 61h \triangleq 'a', ... , 7Ah \triangleq 'z'
- 80h .. FFh : Erweiterung um nat. Sonderzeichen, Blockgrafik und spezielle Symbole (IBM PC-Modelle)

Einige Codeplätze, die in der amerikanischen Referenzversion mit den EDV-typischen Sonderzeichen '[', '\', ']', '{', '|', '}', '~' belegt sind, wurden in der weltweiten Norm für die nationale Anwendung vorbehalten. Die in Deutschland übliche nationale ASCII-Version (DIN 66003) hat diese Plätze mit den Umlauten 'Ü', 'Ö', 'Ä', 'ü', 'ö', 'ä' und 'ß' belegt. Da für jedes ASCII-Zeichen ein Byte verwendet wird, bleibt das 8. Bit b₇ ohne Funktion. Es kann für Prüfzwecke als Paritätsbit (Parity Bit) verwendet werden.

1.4.1 Erweiterung von ASCII im PC-Bereich

Bei Einführung des ersten, PC genannten Mikrocomputers durch die Fa. IBM wurde ein erweiterter ASCII-Zeichensatz festgelegt, der Bit b₇ zur Codierung weiterer 128 Zeichen benutzt. Damit ist es möglich geworden, zusätzlich zu den vorhandenen Zeichen mathematische Symbole, Blockgrafik, nationale Sonderzeichen u.ä. zu verwenden. Da jedoch die Codierungen für die in Deutschland üblichen Umlaute 'Ü', 'Ö', 'Ä', 'ü', 'ö', 'ä' und 'ß' von IBM mit 8 Bit vereinbart wurden, gibt es Unverträglichkeiten, wenn Datenendgeräte wie z.B. Drucker auf 7-Bit ASCII 'German' nach DIN bzw. ISO eingestellt werden.

1.4.2 Der Unicode (ISO 10646)

Der Unicode-Standard (aktuell Version 3.0) ist ein einheitliches Codierungsschema für geschriebene Zeichen und Text. Der Zeichenvorrat dieses für weltweit einheitliche Informationsverarbeitung vorgesehenen Codes umfaßt Zeichen für die Hauptschriften der Welt, aber auch technische Symbole, die sich in allgemeinem Gebrauch befinden. Das Unicode-Codierungsschema (aktuelle Information des Unicode Konsortiums in <http://www.unicode.org/Unicode.charts/normal/Unicode3.0.html>) behandelt alphabetische und ideographische Zeichen sowie Symbole auf die gleiche Weise. Unicode ist vergleichbar dem ASCII-Zeichensatz aufgebaut, benutzt aber zur Codierung 16 Bit. Dabei werden weder Funktionstasten wie die Escape-Taste noch besondere Kontrollsequenzen benötigt, um ein bestimmtes Zeichen in einer bestimmten Sprache anzusteuern.

Mit Unicode sollten zwei Probleme behoben werden:

- Vermeidung einer Überlastung des Fontmechanismus bei multilingualen Computerprogrammen
- Vermeidung von mehrfachen, inkonsistenten Zeichencodes aufgrund unterschiedlicher nationaler und industrieller Zeichenstandards.

Um Kompatibilität mit 7-Bit oder 8-Bit-Umgebungen herzustellen, gibt es die UTF-7 bzw. UTF-8-Standards (UTF bedeutet Universal Text Format), die dafür sorgen, daß Unicode-Codierungen über 7-Bit bzw. 8-Bit orientierte Kommunikationsprotokolle übertragen werden können.

Unicode findet zunehmende Unterstützung bei aktuellen Softwareprodukten, Beispiele: Windows/NT, Netscape Communicator, Word 2000. ES gibt allerdings noch eine Vielzahl älterer Programme, welche nur 8-Bit-Zeichencodes verarbeiten können.

Tabelle 1.9 Konstante und Steuerzeichen in Unicode

Escape-Sequenz	Unicode-Sequenz	Bedeutung	Englischer Name
\b	\u0008	Backspace	backspace
\f	\u000C	Seitenvorschub	form feed
\n	\u000A	Zeilenvorschub	line feed/newline
\r	\u000D	Wagenrücklauf	carriage return
\t	\u0009	horizontaler Tabulator	horizontal tab
\\	\u005C	inverser Schrägstrich	backslash \
\'	\u0027	Apostroph	single quote '
\"	\u0022	Doppelanführungszeichen oben "	double quote

\d		Escape-Sequenz mit	
\dd		hexadezimalen Ziffern	
\cdd	$\leq \text{\u00FF}$	$c \in \{0,..3\}, d \in \{0,..,7\}$	
		Escape-Sequenz mit	
		hexadezimalen Ziffern	
\uxxxx	$\leq \text{\uFFFF}$	$x \in \{0,..9\}, a,..,f,A,..,F\}$	

Tabelle 1.10 Einteilung von Unicode (Auszug, Version 2.0)

Integer-Bereich	Unicode-Bereich	Zeichensatz
0 .. 127	\u0000 .. \u007F	ASCII-Zeichensatz
128 .. 255	\u0080 .. \u00FF	ISO-8859-Latin 1-Zeichensatz
880 ..1023	\u0370 .. \u03FF	griechische Zeichen
1024 .. 1279	\u0400 .. \u04FF	kyrillische Zeichen
1424 .. 1535	\u0590 .. \u05FF	hebräische Zeichen
1536 .. 1791	\u0600 .. \u06FF	arabische Zeichen

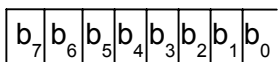
1.5 Parität bei Ziffern- und Zeichencodes

Zur Erkennung von Fehlern bei der Informationsverarbeitung wird häufig die Methode der Paritätsprüfung verwendet. Hierzu muß dem jeweiligen Zeichen ein zusätzliches Prüf- oder Schutzbit hinzugefügt werden, Bei ASCII-Zeichendarstellung wird üblicherweise das unbenutzte 8. Bit verwendet. Damit ergeben sich für häufig benutzte Zeichencodes folgende Zeichenlängen

ASCII nach ISO/DIN: 7 + 1 = 8 Bit
 ASCII (IBM-Erweiterung): 8 + 1 = 9 Bit
 Unicode: 16 + 1 = 17 Bit

Bild 1.3 kennzeichnet die Position des Paritätsbits für die 7-Bit ASCII-Zeichendarstellung. Mit Hilfe von Paritätsbits lassen sich eindeutig nur Einzelbitfehler erkennen, d.h. im Zeichen ist ein einzelnes Bit fehlerhaft

7-Bit-ASCII-Zeichen



gerade Parität : Paritätsbit wird so gewählt, daß die Quersumme über alle Bitstellen (even parity) eine gerade Zahl (0,2,4,...) ergibt.

ungerade Parität : Paritätsbit wird so gewählt, daß die Quersumme über alle Bitstellen (odd parity) eine ungerade Zahl (1,3,5,...) ergibt

Bild 1.3 Festlegung der Parität

Bild 1.4 kennzeichnet am Beispiel der Codierung des Zeichens 'A' (\triangleq 41h) wie das Paritätsbit b7 bei Vereinbarung gerader sowie ungerader Parität gesetzt werden muß.

gerade Parität
(even parity)

0 1 0 0 0 0 0 1 \triangleq 'A' (41h + 00h)



$b_7 = 0$, damit Quersumme gerade bleibt

ungerade Parität
(odd parity)

1 1 0 0 0 0 0 1 \triangleq 'A' (41h + 80h = C1h)



$b_7 = 1$, damit Quersumme ungerade wird

Bild 1.4 Beispiele für die Paritätsfestlegung

Beim Empfang bzw. Weiterverarbeitung der codierten Zeichen wird die jeweilige Quersumme geprüft. Ist das Bildungsgesetz verletzt, so muß ein Übertragungsfehler aufgetreten sein. Einzelbitfehler werden immer erkannt, gleichartige Doppelbitfehler heben sich gegenseitig auf. Eine Fehlererkennung ist dann nicht möglich. Bei sehr hohen Anforderungen an die Zuverlässigkeit der Informationsverarbeitung werden Ziffern- und Zeichencodes mit zusätzlichen Bitstellen verwendet (redundante fehlerkorrigierende Codes). Diese erlauben umfassendere Codeprüfungen und in bestimmten Fällen auch die Korrektur fehlerhafter Codes. Für derartige spezielle Codes ist allerdings ein ganz erheblicher zusätzlicher Aufwand erforderlich. Man verwendet daher meist nur ein Paritätsbit pro Zeichen und fasst eine bestimmte Anzahl von Zeichen zu einem Datenblock zusammen. Hierfür berechnet man jeweils eine Prüfsumme, die auch übertragen bzw. mit gespeichert wird (z.B. CRC oder ECC bei magn. Speichermedien). Bei der Weiterverarbeitung dieser Datenblöcke wird nach dem gleichen Verfahren eine aktuelle Prüfsumme bestimmt und mit der ursprünglichen verglichen. Auf diese Weise lassen mit überschaubarem Zusatzaufwand fehlerhafte Daten erkennen und meist auch korrigieren.. .

1.5.1 Schaltungen für Paritätsbildung und -Prüfung

Für die Bildung des Paritätsbits wird eine logische Schaltung benötigt, die als Paritätsgenerator (parity generator) bekannt ist. Die Prüfung empfangener Zeichen auf die Einhaltung der vorgegebenen Zeichenparität erfolgt mit Hilfe von Paritätsprüfschaltungen (parity checker). Da beide Funktionen logisch sehr ähnlich sind, genügt in der Praxis eine Standardschaltung. Aus der Familie der TTL-Schaltungen wird für diesen Zweck häufig folgende Digitalerschaltung verwendet

74ALS280 9-bit parity generator /checker

Die Kennzeichnung 'ALS' (**a**dvanced **l**ow **p**ower **s**chottky) besagt, daß der Baustein im Vergleich zur TTL-Standardversion kürzere Schaltzeiten besitzt. Er wird u.a. im Mikrocomputerbereich verwendet, um den RAM-Speicher des Computers durch ein zusätzliches 9. Paritätsbit b_8 , das natürlich mitgespeichert werden muß, hardwaremäßig gegen auftretende Speicherfehler zu schützen. Aus diesem Grunde besitzen beispielsweise die RAM-Speicher der verschiedenen PC-Modelle stets die Organisation x9, x18, x36, da pro Speicherbyte ein Paritätsbit erforderlich ist.

1.6 Lerntest/Übungsfragen

1. Welche Vorzüge weisen digitale Systeme gegenüber analogen auf ?
2. Warum werden in der Digitaltechnik nur zwei verschiedene Zustände verwendet ?
3. Wodurch unterscheiden sich digitale von analogen Signalen ?
4. Was versteht man unter einem Code ?
5. Welche Merkmale weisen die wichtigsten Zifferncodes auf ?
6. In welchen Anwendungsbereichen werden einschriftige Codes verwendet ?
7. Wieviel Bits werden bei der ASCII-Zeichendarstellung benötigt ?
8. Wodurch unterscheiden sich ASCII- und Unicode ?
9. Was versteht man unter "even parity" ?
10. Was ist ein "parity checker/generator" ?
11. Welche Fehlerarten lassen sich durch ein P-Bit erkennen ?
12. Wie können Mehrbitfehler erkannt werden ?