

---

Seminar SS 2010 – Mobile Computing  
11.12.2010

# Einführung in QR Codes

Martin Stoev (inf8928)



# Motivation

---

- ▶ QR Code verwendet, aber was passiert genau?
- ▶ Wie funktionieren QR Codes?
- ▶ Welche Algorithmen und Techniken werden eingesetzt?
  - ▶ Durchstich durch alle verwendeten Methoden
  - ▶ Beispiele für besseres Verständnis
- ▶ Ziel: Grundlagen nachvollziehen können, aber rechnen soll Computer
- ▶ Wie kann der Standard an eigene Bedürfnisse angepasst werden? – Konkret: Mobile Computing

# Agenda

---

- ▶ **Einleitung**
- ▶ Grundbegriffe
- ▶ Informationen in QR Code Symbolen
- ▶ Funktionen und Algorithmen
  - ▶ Masking
  - ▶ Fehlerkorrektur
  - ▶ Encoding (mit Beispiel)
  - ▶ Decoding
- ▶ Erweiterungsmöglichkeiten
- ▶ Fazit

# Was ist ein QR Code?

---

- ▶ QR Code = Quick Response Code
- ▶ 2-D Symbol: 3 charakteristische Finder Patterns
- ▶ Erfunden: 1994 von Denso (Toyota)
- ▶ Standard: ISO/IEC 18004 (2000 bzw. 2006)
- ▶ Primäres Ziel: Automatisierung und Kontrolle der Produktion von Autobauteilen
- ▶ Verschiedene Modelle: **Model 2**, Micro QR Code, ...
- ▶ Generell: Speichern digitaler Daten in analoger Form (digital → analog → digital)
  - ▶ Encoding: digital → analog
  - ▶ Decoding: analog → digital

<http://martinstoev.de> →



→ <http://martinstoev.de>

Abb.: QR Code (digital → analog → digital)

# Hauptfeatures

---

- ▶ Schnelles Decoding (unabhängig von Orientierung)
- ▶ Robust / Tolerant
  - ▶ Korrektur bei Verformungen
  - ▶ Fehlerkorrektur beschädigter Symbole (bis 30% Fehlerrate)
- ▶ Masking (charakteristische Muster, optimale Erkennung garantiert)
- ▶ Verschiedene Zeichensätze (erweiterbar)



**Abb.:** Rotiert



**Abb.:** Verzerrt



**Abb.:** Verschmutzt

# QR Code im Vergleich

---

## ▶ 2-D Codes vs 1-D Codes

- ▶ Höhere Datendichte
- ▶ Schlechtere Erkennung, da Muster komplexer

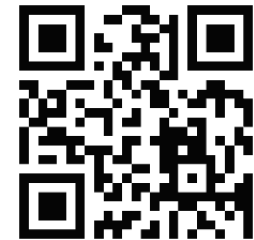
## ▶ QR Codes vs 2-D Codes

- ▶ Schnellere Erkennung durch besondere Muster
  - ▶ Position im Bild (Von wo bis wo?)
  - ▶ Orientierung (Drehung, Winkel)
- ▶ Leicht erweiterbar (Abweichung vom Standard)
- ▶ Standard ohne Lizenzkosten



martinstoev.de

**Abb.:** Code-128 1-D Barcode



**Abb.:** QR Code

# Mobile Computing

---

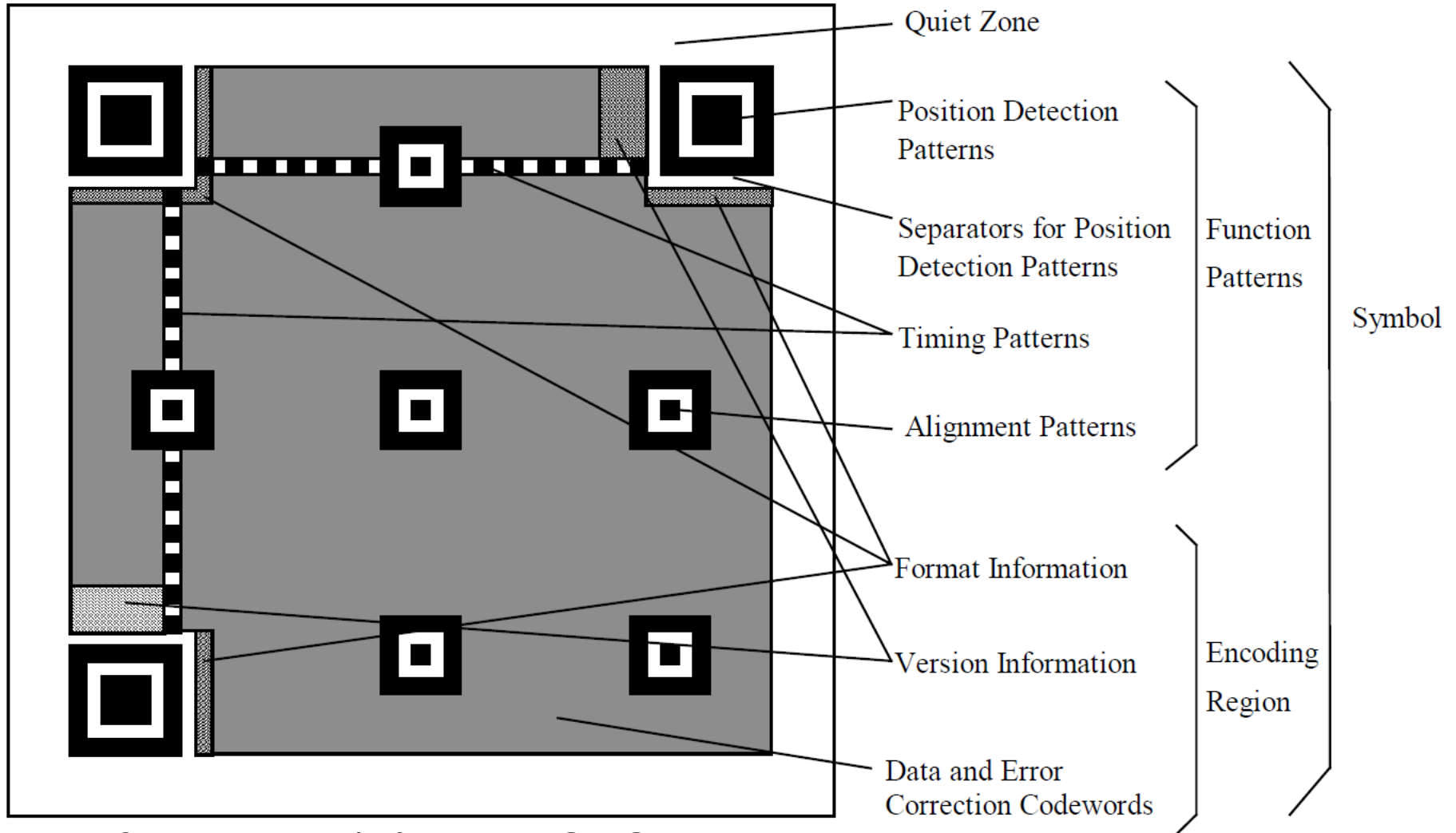
- ▶ Kamera und (kostenlose) Software bei Mobilgeräten weit verbreitet
  - ▶ Quickmark Reader
  - ▶ BeeTagg Reader
  - ▶ Kaywa Reader
  - ▶ Neo Reader (...)
- ▶ Generatoren: z.B. <http://zxing.appspot.com/generator/>
- ▶ Prozess: Daten decodieren und mit korrekter Applikation aufrufen
- ▶ Geringer Aufwand für Nutzer: fotografieren statt abschreiben
- ▶ Orientierung des QR Code bei Aufnahme irrelevant

# Agenda

---

- ▶ Einleitung
- ▶ **Grundbegriffe**
- ▶ Informationen in QR Code Symbolen
- ▶ Funktionen und Algorithmen
  - ▶ Masking
  - ▶ Fehlerkorrektur
  - ▶ Encoding (mit Beispiel)
  - ▶ Decoding
- ▶ Erweiterungsmöglichkeiten
- ▶ Fazit



# Struktur eines QR Code Symbols



**Abb.:** Schematischer Aufbau eines QR Code Version 7

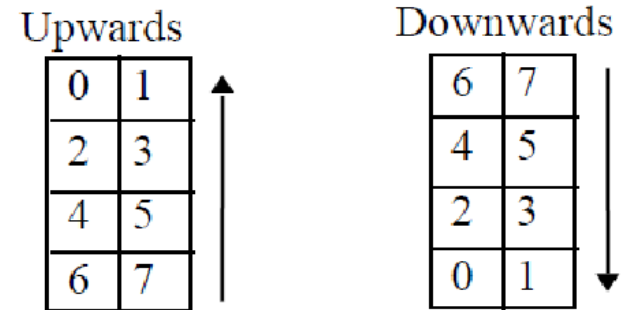
# Modul

---

- ▶ Kleinste Einheit in QR Code Symbol
- ▶ 2 Ausprägungen
  - ▶ dunkel: binäre 1 
  - ▶ hell: binäre 0 
- ▶ Direkte Konvertierung: Bit  $\hat{=}$  Daten Modul
- ▶ Daten und Fehlerkorrekturmodule werden in Codewörtern (8 Module) abgelegt

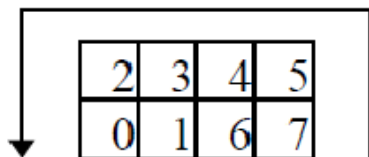
# Codewort

- ▶ Codewort  $\hat{=}$  Daten Byte, besteht aus 8 Modulen
- ▶ Anordnung der Module hängt von Richtung ab
- ▶ Höchstwertigstes Modul: 7
- ▶ Regeln bei Begrenzungen:
  - ▶ Bestmöglich umfließen
  - ▶ Codewort in 2 Teile zerlegen

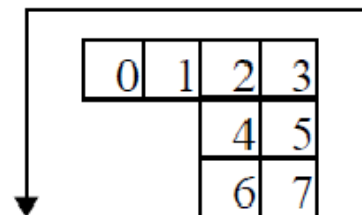


**Abb.:** Fundamentale Richtungen

Upwards to Downwards (i)

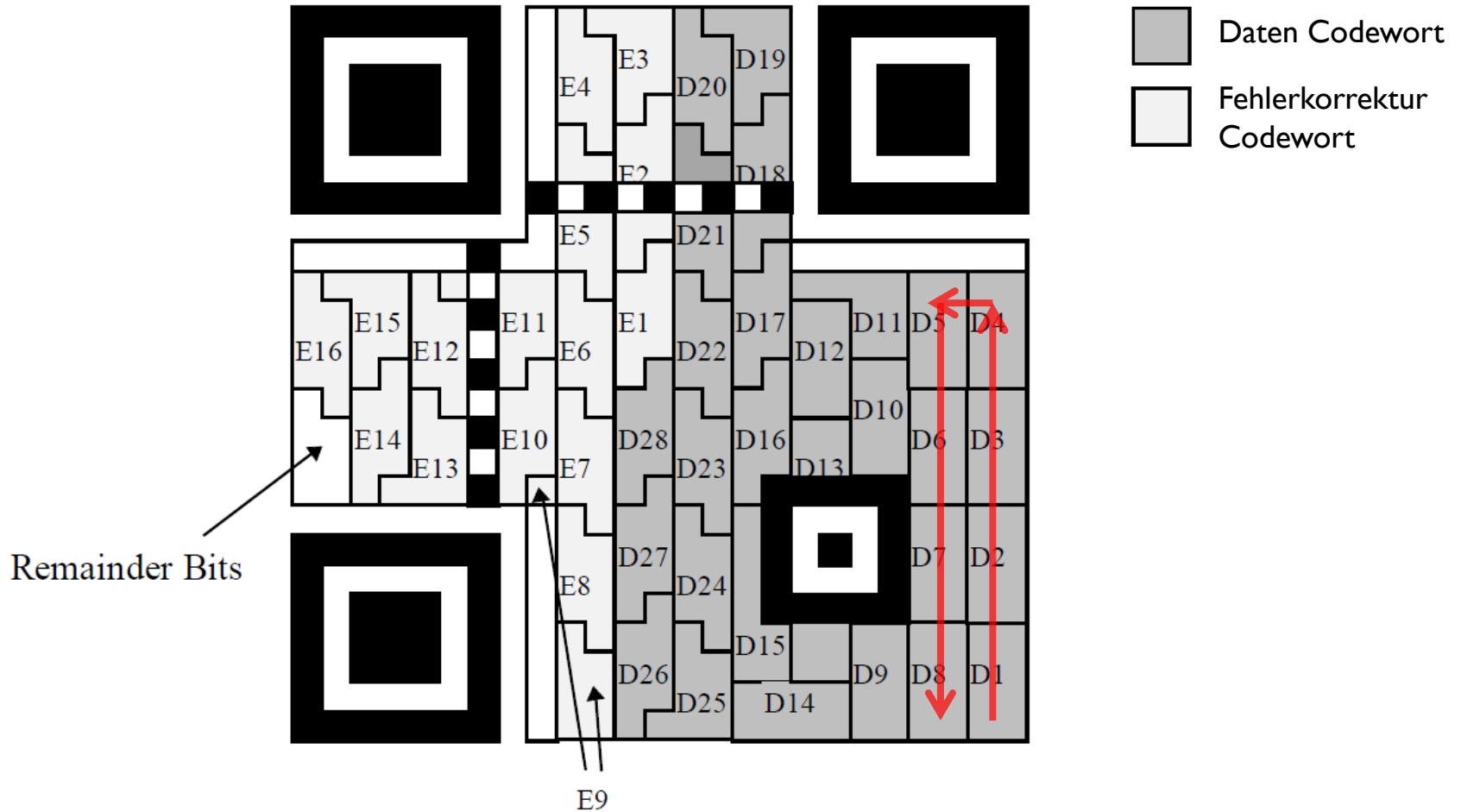


Upwards to Downwards (ii)



**Abb.:** Fließrichtungen bei Grenzen

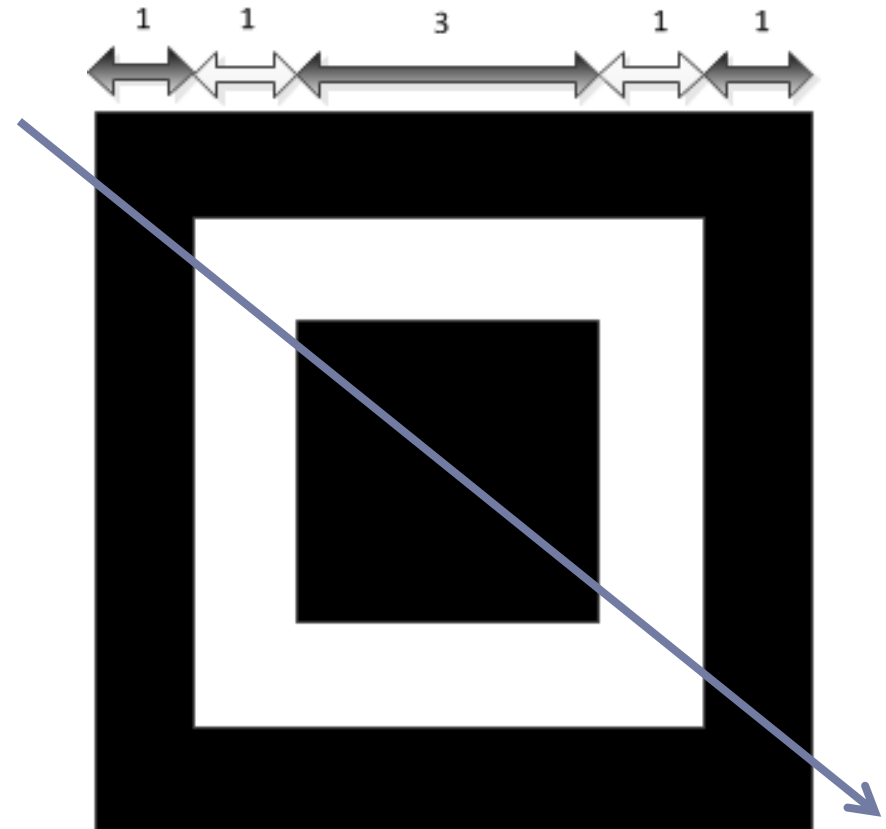
# Codewörter in einem QR Code Symbol



**Abb.:** Schematische Darstellung der Codewörter eines 2-M QR Codes

# Finder Pattern

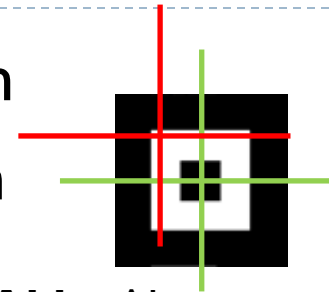
- ▶ Ermöglicht das Erkennen von QR Codes
  - ▶ Exakt 3 Finder Patterns in QR Code
  - ▶ Muster wird in Daten unterdrückt
- ▶ Einheitliche Form
  - ▶ Quadratisch, umrandet von Separator (helle Module)
  - ▶ Verhältnis von hellen und dunklen Modulen: 1:1:3:1:1 (unabhängig von Rotationswinkel)
  - ▶ Definiert indirekt Modulgröße



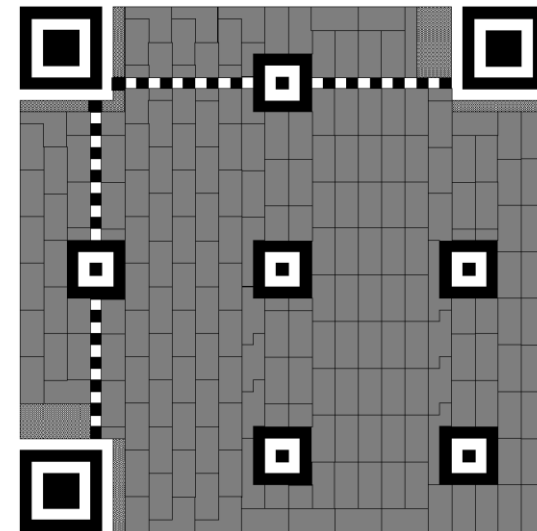
**Abb.:** Finder Pattern mit charakteristischen Abständen 1:1:3:1:1

# Alignment Pattern

- ▶ Verzerrungen und Verformungen korrigieren
- ▶ Berechnete Position vs tatsächliche Position
- ▶ Fest definierte Positionen
  - ▶ Dürfen Finder Patterns nicht überlagern
  - ▶ Dürfen in Timing Pattern integriert sein



**Abb.:** Alignment Pattern



**Abb.:** Alignment Pattern  
Positionen in Version 7 Symbol

Version	#	Positionen						
1	0	-						
2	1	6	18					
3	1	6	22					
...								
7	6	6	22	38				
...								
40	46	6	30	58	86	114	142	170

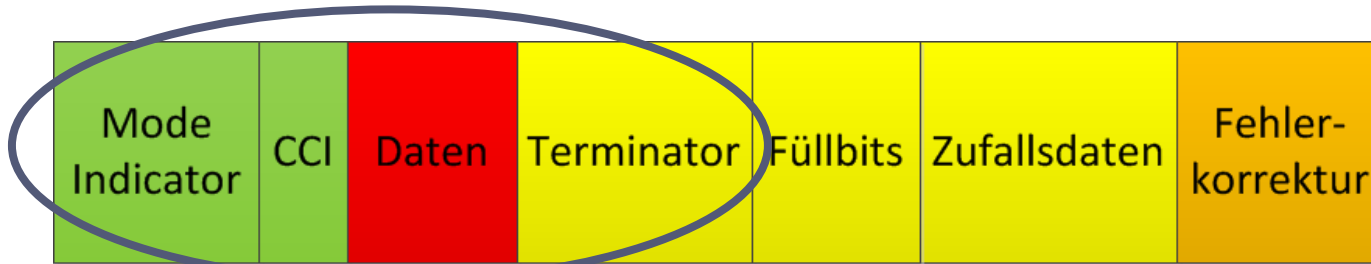
**Tab.:** Positionen der Alignment Pattern

# Agenda

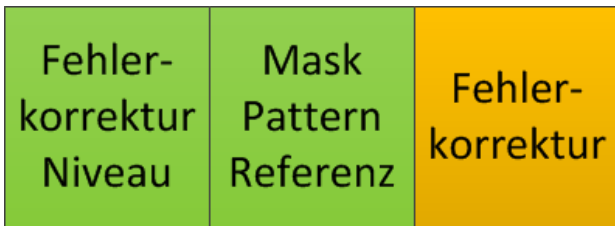
---

- ▶ Einleitung
- ▶ Grundbegriffe
- ▶ **Informationen in QR Code Symbolen**
- ▶ Funktionen und Algorithmen
  - ▶ Masking
  - ▶ Fehlerkorrektur
  - ▶ Encoding (mit Beispiel)
  - ▶ Decoding
- ▶ Erweiterungsmöglichkeiten
- ▶ Fazit

# Informationen in QR Codes



Daten in Codewörtern



Formatinformationen



Versionsinformationen



Metadaten



Daten



Redundant



Daten für Fehlerkorrektur

Version	Fehlerkorrektur
---------	-----------------

# Versionsinformation

- ▶ Version gibt die Größe des QR Code Symbol an
- ▶ Es gibt 40 unterschiedliche Versionen
  - ▶ Version 1: 21x21 Module
  - ▶ Version 40: 177x177 Module
  - ▶ Pro Version wächst Dimension jeweils um 4 Module:
- ▶ 18 Modul Sequenz: 6 Module für Version, 12 für Fehlerkorrektur
- ▶ Versionsinformation
  - ▶ Version  $\leq 6$ : Nicht vorhanden  $\rightarrow$   $Version = \frac{Breite}{Dimension} - 10$   
4
  - ▶ Version  $> 6$ : berechnen, wie im Beispiel
- ▶ Beispiel
  - ▶ Versionsnummer: 7
  - ▶ Binärdaten: 000111
  - ▶ Fehlerkorrekturdaten: 110010010100
  - ▶ Formatinformation: 000111110010010100
- ▶ Redundant in Symbol gespeichert (2 Mal)

# Version

Version	Fehlerkorrektur Niveau	Anzahl Daten Codewörter	Anzahl Bits	Datenkapazität	
				Numeric	Alphanumeric
1	L	19	152	41	25
	M	16	128	34	20
	Q	13	104	27	16
	H	9	72	17	10
2	L	34	272	77	47
	M	28	224	63	38
	Q	22	176	48	29
	H	16	128	34	20
3	L	55	440	127	77
	M	44	352	101	61
	Q	34	272	77	47
	H	26	208	58	35
...					
40	L	2.956	23.648	7.089	1.817
	M	2.334	18.672	5.596	1.435
	Q	1.666	13.328	3.993	1.024
	H	1.276	10.208	3.057	784

**Tab.:** Versionen und ihre Kapazitäten

# Formatinformationen

Fehlerkorrektur Niveau	Mask Pattern Referenz	Fehlerkorrektur
------------------------	-----------------------	-----------------

- ▶ 15 Modul Sequenz: 5 Module für Format, 10 für Fehlerkorrektur
  - ▶ 2 Module Fehlerkorrektur Niveau
  - ▶ 3 Module Mask Pattern
- ▶ XOR mit festgelegter Maske (101010000010010)
- ▶ Beispiel
  - ▶ Fehlerkorrektur Niveau M: 00
  - ▶ Mask Pattern Referenz: 101
  - ▶ Format: 00101
  - ▶ Fehlerkorrekturdaten: 0011011100
  - ▶ Unmaskierte Formatinformationen: 001010011011100
  - ▶ Festgelegte Maske: 101010000010010
  - ▶ Formatinformationen: 100000011001110
- ▶ Redundant in Symbol gespeichert (2 Mal)

# Modus

---

- ▶ Interpretation der Daten → Zeichensatz
- ▶ 4 Modi
  - ▶ Numeric Mode: [0-9]
  - ▶ Alphanumeric Mode: [0-9A-Z\$%\*+-./:] + Leerzeichen
  - ▶ 8-Bit Byte Mode: Daten aller Art
  - ▶ Kanjii Mode: Japanischer Zeichensatz
- ▶ ECI: Extended Channel Interpretation (wechseln des Modus)

Mode	Indicator
ECI	0111
Numeric	0001
Alphanumeric	0010
...	
Terminator (End of Message)	0000

**Tab.:** Mode Indicator Tabelle

# Numeric Mode

---

- ▶ **Ziffernfolge in Blöcke mit 3 Ziffern zerlegen**
  - ▶ Block als Zahl interpretieren → 10 Bit Folge
  - ▶ Ausnahme: Letzte Bits → 7 Bit Folge oder 10 Bit Folge
- ▶ **Beispiel**
  - ▶ Ziffernfolge: 01234567 → (012, 345, 67)
  - ▶ 012 → 0000001100
  - ▶ 345 → 0101011001
  - ▶ 67 → 1000011
  - ▶ Resultat: 000000110001010110011000011

# Agenda

---

- ▶ Einleitung
- ▶ Grundbegriffe
- ▶ Informationen in QR Code Symbolen
- ▶ **Funktionen und Algorithmen**
  - ▶ Masking
  - ▶ Fehlerkorrektur
  - ▶ Encoding (mit Beispiel)
  - ▶ Decoding
- ▶ Erweiterungsmöglichkeiten
- ▶ Fazit

# Masking

- ▶ Ziel: Bestmögliche Erkennung eines QR Code
  - ▶ Unterdrücken von Flächen gleicher Module
  - ▶ Unterdrücken von Finder Patter
  - ▶ Optimal schwarz/weiß rauschen
- ▶ Erstellen einer XOR Maske
  - ▶ Funktion:  $f(x,y) \rightarrow \text{Boolean}$ 
    - ▶ True: Modul flippen
    - ▶ False: nichts machen
    - ▶ Nur Daten maskieren (nicht: Finder, Alignment und Timing Patterns oder Metainformationen)
    - ▶ 8 Funktionen (Mask Pattern) stehen zur Verfügung
  - ▶ Optimale Maske mit festgelegten Bewertungskriterien finden
    - ▶ Alle Funktionen ausprobieren und Strafpunkte aufsummieren
    - ▶ Funktion mit geringsten Strafpunkten auswählen

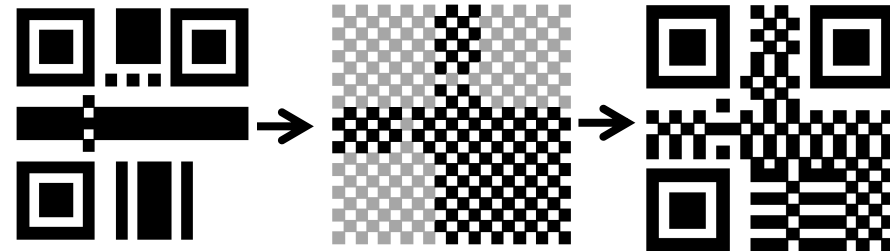


Abb.: Unmaskiertes Symbol XOR Maske  $\rightarrow$  QR Code

# Mask Pattern

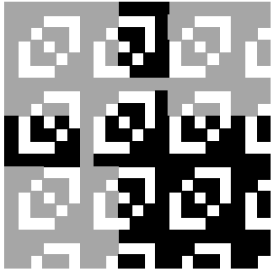


Abb.: I10 Mask Pattern

Mask Pattern Referenz	Bedingung
000	$(i + j) \bmod 2 = 0$
001	$i \bmod 2 = 0$
010	$j \bmod 3 = 0$
011	$(i + j) \bmod 3 = 0$
100	$((i \text{ div } 2) + (j \text{ div } 3)) \bmod 2 = 0$
101	$(i \bmod 2) + (j \bmod 3) = 0$
110	$((i \bmod 2) + (j \bmod 3)) \bmod 3 = 0$
111	$((i \bmod 3) + (i + j) \bmod 2) \bmod 2 = 0$

Tab.: Mask Pattern Referenzen zu Masking Funktionen

Eigenschaft	Bewertungskriterium	Punkte
Hintereinander liegende Module in Zeile / Spalte haben die gleiche Farbe	Anzahl der gleichfarbigen Module = $(5 + i)$	$3 + i$
Block mit Modulen in gleicher Farbe	Blockgröße = $m * n$	$3 * (m - 1) * (n - 1)$
1:1:3:1:1 Muster in Zeile / Spalte		40
Verhältnis helle zu dunkle Module im gesamten Symbol	$50 (5^k)\%$ zu $50 \pm (5^*(k+1))\%$	$10 * k$

Tab.: Bewertungskriterien mit Strafpunkten

# Fehlerkorrektur Tabellen

Version	Anzahl Codewörter insgesamt	Fehlerkorrektur Niveau	Anzahl Fehlerkorrektur Codewörter	Anzahl der Fehlerkorrektur Blocks	Fehlerkorrektur pro Block (c,k,r)
1	26	L	7	1	(26,19,2)
		M	10	1	(26,16,4)
		Q	13	1	(26,13,6)
		H	17	1	(26,9,8)
2	44	L	10	1	(44,34,4)
		M	16	1	(44,28,8)
		Q	22	1	(44,22,11)
		H	28	1	(44,16,14)
3	70	L	15	1	(70,55,7)
		M	26	1	(70,44,13)
		Q	36	2	(35,17,9)
		H	44	2	(35,13,11)
...					
40	3.532	L	750	19 6	(147,117,15) (148,118,15)
		M	1.372	18 31	(75,47,14) (76,48,14)
		Q	2.040	34 34	(54,24,15) (55,25,15)
		H	2.430	20 61	(45,15,15) (46,16,15)

(c,k,r):  
 c = Anzahl Codewörter insgesamt,  
 k = Anzahl Daten Codewörter,  
 r = Fehlerkorrekturkapazität

Fehlerkorrektur Niveau	Indikator	Fehlerkorrektur rate (%)
L	01	7
M	00	15
Q	11	25
H	10	30

**Abb.:** Fehlerkorrektur Niveau

# Fehlerkorrektur

---

- ▶ Reed-Solomon Codes mit Erzeugerpolynom
- ▶ Rechnen mit Koeffizienten von Polynomen
  - ▶ Polynomdivision
  - ▶ Gleichungssysteme
  - ▶ Symbolalphabet fester Größe (Galois Feld)
- ▶ Einsatz in bekannten Standards: CD, DVB, RS Codes, usw.
- ▶ Fehlerkorrekturrate festlegbar
  - ▶  $2t$  Fehlerkorrektursymbole können garantiert  $t$  Fehler korrigieren
  - ▶ Auch Burstfehler
  - ▶ Problem: Auch Korrektursymbole können Fehler enthalten

# RS Encoding (1)

- ▶ GF(8) mit Primitiv Polynom  $X^3 + X + 1$  erzeugen
- ▶ Erzeugerpolynom berechnen
  - ▶ 2 Fehler korrigieren  $\rightarrow$  4 Symbole
  - ▶ Einheitswurzeln  $a^0 - a^3 \rightarrow g(X)$

$$g(X) = (X + a^0)(X + a^1)(X + a^2)(X + a^3)$$

$$g(X) = X^4 + a^2X^3 + a^5X^2 + a^5X + a^6$$

(ausmultipliziert)

Element	Polynom
0	000
$a^0$	001
$a^1$	010
$a^2$	100
$a^3$	011
$a^4$	110
$a^5$	111
$a^6$	101

**Tab.:** GF (8)

- ▶ Koeffizientendarstellung:  $a^0 a^2 a^5 a^5 a^6$
- ▶ Entsprechende Tabellen mit Erzeugerpolynomen in ISO 18004 definiert

# RS Encoding (2)

- ▶ Datensequenz: 111001111  $\rightarrow a^5 a^0 a^5$
- ▶ Fehlerkorrektursymbole

$$a^5 a^0 a^5 0 0 0 0 : a^0 a^2 a^5 a^5 a^6 = a^5 0 a^2 \text{ Rest } a^6 a^5 a^0 a^1$$

$$\begin{array}{r} a^5 a^0 a^3 a^3 a^4 \\ \hline a^2 a^3 a^4 0 0 \\ a^2 a^4 a^0 a^0 a^1 \\ \hline a^6 a^5 a^0 a^1 \end{array}$$

Element	Polynom
0	000
$a^0$	001
$a^1$	010
$a^2$	100
$a^3$	011
$a^4$	110
$a^5$	111
$a^6$	101

Tab.: GF (8)

- ▶ Daten Codewörter + Fehlerkorrektur Codewörter
- ▶  $a^5 a^0 a^5 a^6 a^5 a^0 a^1 = 111001111101111001010$

# RS Decoding(1)

---

- ▶ Daten empfangen:

$$R = (r_0, r_1, r_2, r_3, r_4, r_5, r_6)$$

$$\text{bzw. } R(x) = (r_0, r_1x, r_2x^1, r_3x^2, r_4x^3, r_5x^4, r_6x^5)$$

- ▶ Syndrom berechnen:

$$S_0 = R(1) = r_0 + r_1 + r_2 + r_3 + r_4 + r_5 + r_6$$

$$S_1 = R(\alpha) = r_0 + r_1\alpha + r_2\alpha^2 + r_3\alpha^3 + r_4\alpha^4 + r_5\alpha^5 + r_6\alpha^6$$

$$S_2 = R(\alpha^2) = r_0 + r_1\alpha^2 + r_2\alpha^4 + r_3\alpha^6 + r_4\alpha^1 + r_5\alpha^3 + r_6\alpha^5$$

$$S_3 = R(\alpha^3) = r_0 + r_1\alpha^3 + r_2\alpha^6 + r_3\alpha^2 + r_4\alpha^5 + r_5\alpha^1 + r_6\alpha^4$$

- ▶ Fehlerpositionen finden ( $\sigma_1$  und  $\sigma_2$  ermitteln):

$$S_0\sigma_2 + S_1\sigma_1 + S_2 = 0$$

$$S_1\sigma_2 + S_2\sigma_1 + S_3 = 0$$

- ▶ Alle Elemente aus GF(8) einsetzen:  $\sigma(x) = \sigma_2 + \sigma_1x + x^2$

- ▶ Fehler an Position  $j$ , wenn  $\sigma(\alpha_j) = 0$

## RS Decoding (2)

---

- ▶ Fehlergröße  $Y_1$  und  $Y_2$  bestimmen:

$$Y_1 \alpha^j + Y_2 \alpha^{2j} = S_0$$

$$Y_1 \alpha^{2j} + Y_2 \alpha^j = S_1$$

- ▶ Fehler korrigieren: Komplement der Fehlergröße zu jeder Fehlerposition addieren

# QR Encoding

---



1. Daten encodieren
2. Fehlerkorrektur Codewörter berechnen
3. Anfangsmatrix erstellen
4. Masking Pattern auswählen
5. Format Informationen hinzufügen
6. QR Code generieren

# Encoding Beispiel



- ▶ Encoding der Ziffernfolge: 01234567
- ▶ Fehlerkorrektur Niveau: M  $\rightarrow$  00
- ▶ Modus: Numeric Mode  $\rightarrow$  Mode Indicator: 0001
- ▶ CCI: 8  $\rightarrow$  Version: I-M ( $34 \geq 8$ ) mit 16 CW
- ▶ Versionsinformation: keine, da Version  $< 7$
- ▶ Daten: 01234567  $\rightarrow$  (012, 345, 67)  $\rightarrow$   
000000110001010110011000011
- ▶ Terminator: 0000
- ▶ Mode Indicator + CCI + Daten + Terminator + Füllbits (6 CW):  
00010000 00100000 00001100 01010110 01100001 10000000
- ▶ Zufallsdaten (10 CW): 11101100 00010001 11101100  
00010001 11101100 00010001 11101100 00010001 11101100  
00010001



# Encoding (Fehlerkor.)

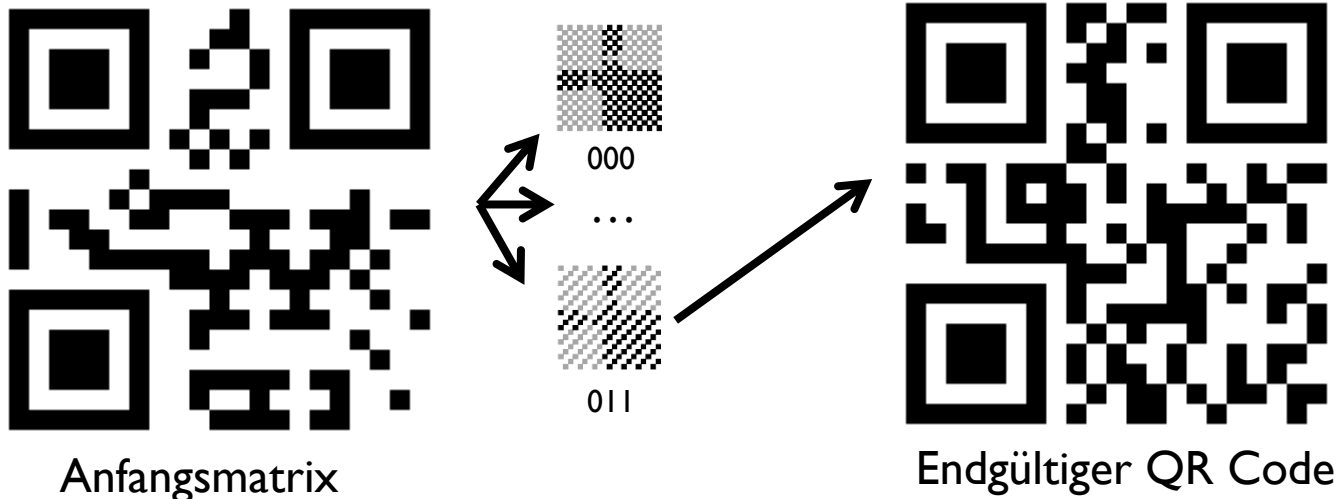


- ▶ Fehlerkorrektur (10 CW): 10100101 00100100 11010100  
11000001 11101101 00110110 11000111 10000111  
00101100 01010101
- ▶ Alle Codewörter (26 CW): 00010000 00100000  
00001100 01010110 01100001 10000000 11101100  
00010001 11101100 00010001 11101100 00010001  
11101100 00010001 11101100 00010001 10100101  
00100100 11010100 11000001 11101101 00110110  
11000111 10000111 00101100 01010101

# Encoding (Anfangsmatrix)

---

- ▶ Bitfolge in Anfangsmatrix einfügen
- ▶ Alle Mask Patterns XOR Anfangsmatrix
- ▶ Strafpunkte addieren, vergleichen und Masking Referenz mit geringster Punktzahl auswählen: 011
- ▶ Formatinformationen berechnen: 00011



# Encoding (Formatinformation)

Fehlerkorrektur Niveau	Mask Pattern Referenz	Fehlerkorrektur
------------------------	-----------------------	-----------------

- ▶ Fehlerkorrektur Niveau:  $M \rightarrow 00$
- ▶ Masking Pattern Referenz:  $011$
- ▶ Fehlerkorrektur:  $1101011001$
- ▶ Vorläufige Formatinformation:  $000111101011001$
- ▶ XOR  $101010000010010$  (festgelegtem Bitmuster)
- ▶ Formatinformation:  $101101101001011$



Endgültiger QR Code

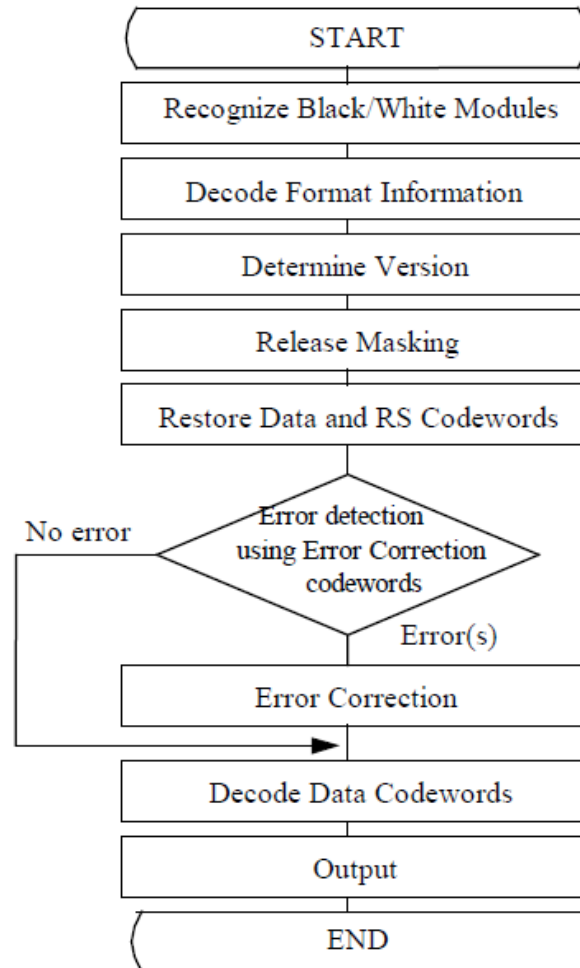
# Decoding

---

- ▶ QR Code liegt entweder als Bild vor oder muss fotografiert werden
- ▶ Während der Erkennungsphase wird der QR Code
  - ▶ Gefunden
  - ▶ Normalisiert
  - ▶ Digitalisiert
    - ▶ Dunkle Felder werden in 1 konvertiert
    - ▶ Helle Felder werden in 0 konvertiert
- ▶ Nach der Erkennungsphase wird das Bild nicht mehr benötigt

# Decoding

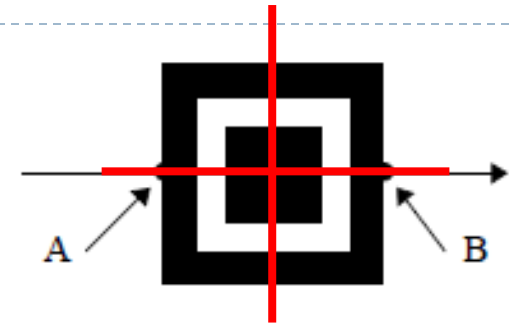
---



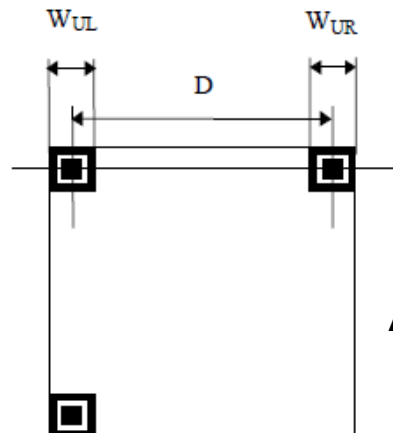
**Abb.:** Flussdiagramm Decoding

# Decoding Pseudocode (1)

- ▶ Kontrast erhöhen
- ▶ Finder Pattern finden
- ▶ Orientierung feststellen und korrigieren
- ▶ Breite  $W_{UL}$ ,  $W_{UR}$  und  $D$  feststellen
- ▶ Modulgröße berechnen  $X = \frac{(W_{UL} + W_{UR})}{14}$
- ▶ Vorläufige Version berechnen  $V = \frac{\frac{D}{X} - 10}{4}$ 
  - ▶  $V < 7 \rightarrow$  Version übernehmen
  - ▶  $V \geq 7 \rightarrow$  Modulgröße berechnen, Versionsinformation auslesen und korrigieren



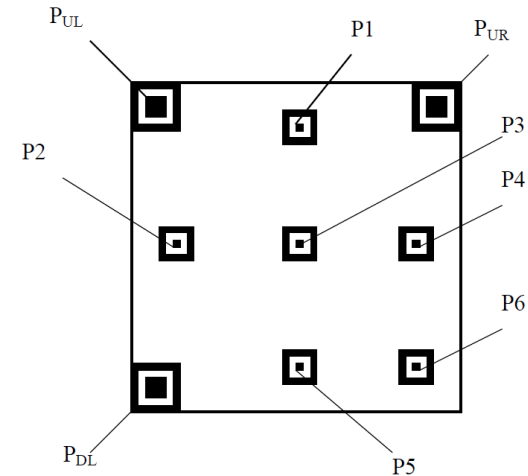
**Abb.:** Finder Pattern finden



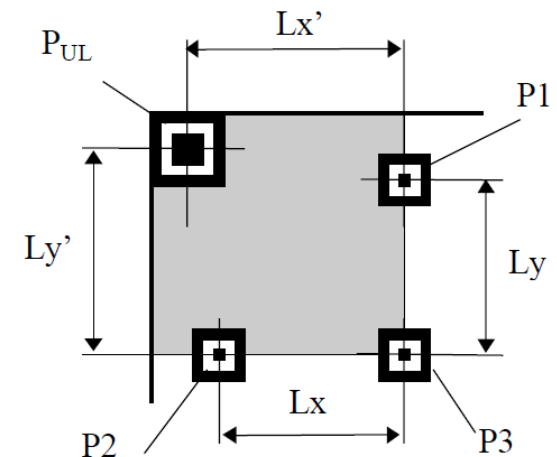
**Abb.:** Distanzen

# Decoding Pseudocode (2)

- ▶ Flächen entzerren (Version > 1)
  - ▶ Position von Alignment Patterns feststellen
  - ▶ Positionen der Alignment Patterns berechnen
  - ▶ Tatsächliche Positionen feststellen
  - ▶ Abweichung berechnen
  - ▶ Immer für 4 Fixpunkte die Fläche entzerren
  - ▶ Anfangen mit Finder Patterns und Alignment Patterns entlang des Timing Patterns



**Abb.:** Reihenfolge Alignment Patterns



**Abb.:** Fläche entzerren

# Decoding Pseudocode (3)

---

- ▶ Binärmatrix erzeugen (Module auslesen)
- ▶ Formatinformationen auslesen
- ▶ Mask Pattern rückgängig machen (Masking Referenz)
- ▶ Codewörter sequentiell mit Richtungsregeln extrahieren
- ▶ Codewörter in Blöcke unterteilen
- ▶ Fehlerkorrektur durchführen
- ▶ Ursprünglichen Bitstrom wiederherstellen
- ▶ Bitstrom in Informationssegmente (Mode Indicator, CCI, ...) aufteilen
- ▶ Daten anhand der Regeln für Mode Indicator decodieren

# Erweiterungsmöglichkeiten (neuer Mode)

---

- ▶ Einfügen eines neuen Mode
  - ▶ Ablegen der Referenz in Tabelle
  - ▶ Alphabet festlegen
  - ▶ Interpretationsregeln implementieren
- ▶ Vorteil: Definieren eines eigenen Alphabets → Effizienz
- ▶ Nachteil: Nur eigene Software kann QR Code decodieren
- ▶ In Praxis eher selten

# Erweiterungsmöglichkeiten (Prozess)

---

- ▶ Decodierungsprozess erweitern
- ▶ Nach der Decodierung die Daten analysieren
  - ▶ Auf Muster matchen (`http://.*`, `mailto://.*`, WLAN)
  - ▶ Eigene Präfixes / Magic Numbers einfügen
- ▶ **Vorteile:**
  - ▶ Weniger Aufwand für Benutzer
  - ▶ Falschinterpretation durch Benutzer unterbunden
  - ▶ Unabhängig vom Standard
- ▶ **Nachteile:**
  - ▶ Platzverschwendung
  - ▶ nur Heuristik, Fehler möglich
- ▶ Wird in der Praxis häufig eingesetzt

# Fazit

---

- ▶ Encoding ist leichter als Decoding
- ▶ Interpretation des Inhaltes von Symbolen im Standard nicht definiert → Selbst implementieren
- ▶ Sinnvoll, wenn digitale Daten kostengünstig analog gespeichert werden müssen
- ▶ Obwohl Standard offen ist, lohnt sich eine neue Implementation von QR Code Software nicht
- ▶ Gut geeignet für mobile Geräte

# Fragen

---

- ▶ Was sind die Hauptfeatures von QR Codes?
  - ▶ Schnelles Decoding, Robust / Tolerant, Masking Prozess, Verschiedene Zeichensätze
- ▶ Lohnt es sich kurze Texte wie „Hallo Welt!“ in QR Codes abzulegen? Wieso?
  - ▶ Nein – Besser Text direkt darstellen
- ▶ Lohnt es sich WLAN Zugangsinformationen in QR Codes abzulegen? Wieso?
  - ▶ Ja – Benutzer muss Daten nicht abschreiben → weniger Fehler
- ▶ Können dynamische Inhalte in QR Codes gespeichert werden? Wie?
  - ▶ Nicht direkt – serverbasierte Lösung möglich (z.B. URL)

# Fragen und Antworten

---

?!

# Quellen

---

- ▶ ISO/IEC 18004:2000
- ▶ Synthesis Journal 2008 - QR Code (S. 59 – 78)
- ▶ Peter Sweeny – Error Control Coding (Kapitel 7 Reed Solomon Codes)