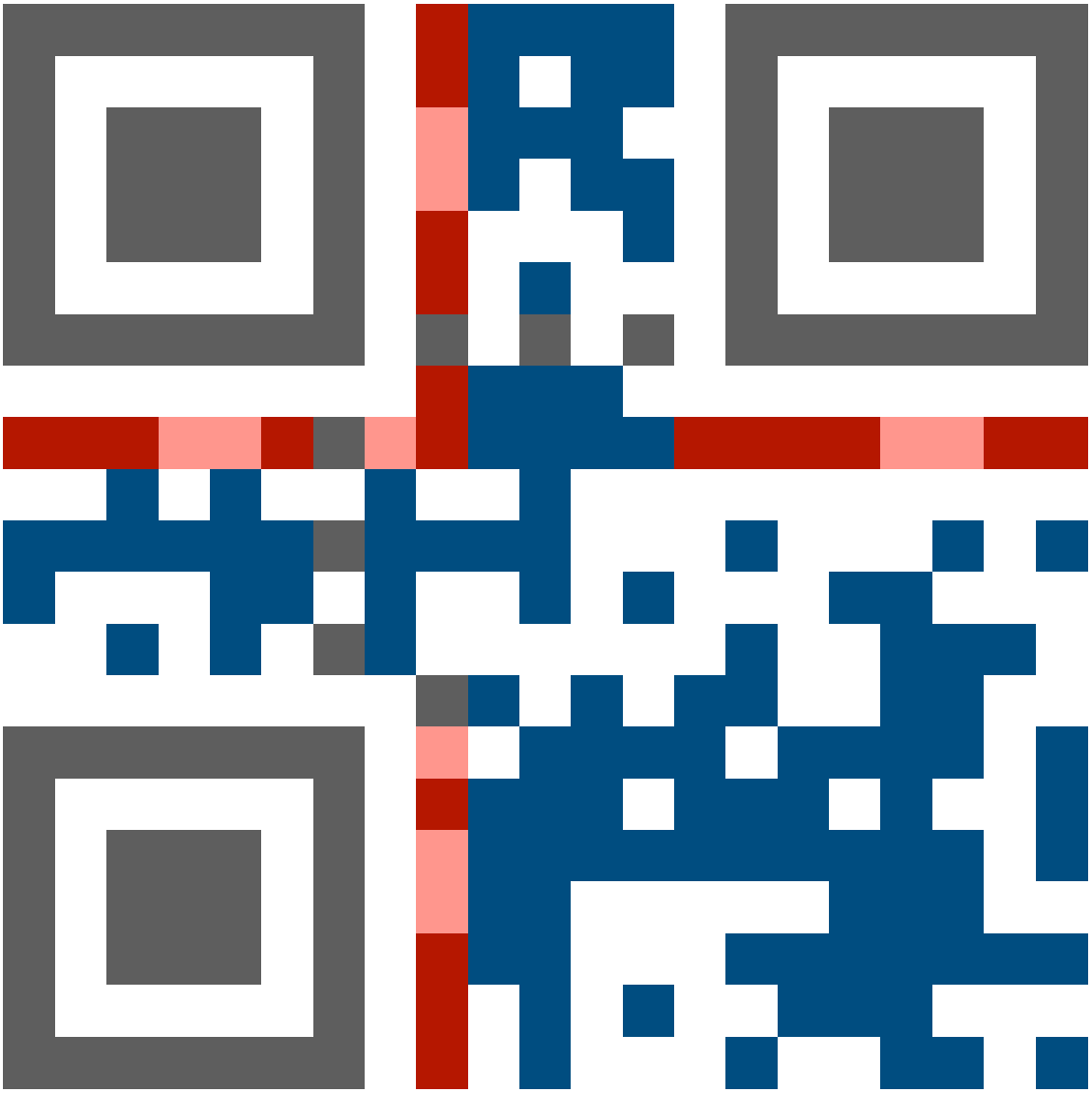
## Praktische Übung: der unbekannte Code

Wir finden den folgenden QR-Code. Die festen Elemente wie Positionsmarker etc. sind bereits grau markiert und die vier Stellen mit der Formatinformation sind rot hinterlegt. Damit bleibt nur der Datenbereich übrig, der bei gesetzten Bits blau gefärbt ist.

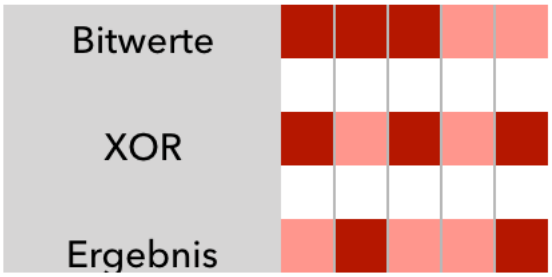


## Was sind die nötigen Schritte?

- Fehlerkorrektur und Maske aus der Formatinfo lesen
- Maske anwenden und die betroffenen Bits invertieren
- Datentyp und Datenlänge lesen
- Bits auslesen und konvertieren

## Feststellen von ECC und Maskencode

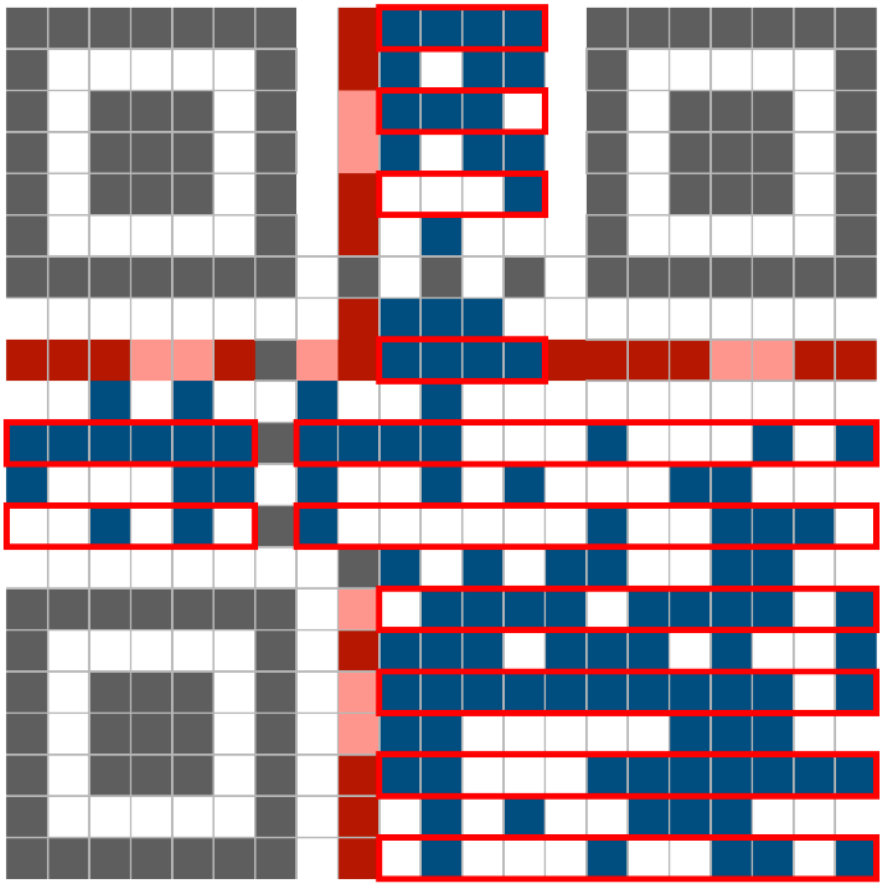
Die ersten fünf Bits der Formatinfo ansehen und mit `10101` per XOR verknüpfen.




Damit erhalten wie `01` für den EC-Code und `001` für die Nummer der Maske => Maske 1

Die besitzt die Formel `i % 2 = 0` (die Zeilennummer ist ohne Rest durch 2 teilbar). Diese Maske legen wir nun über den Code. Damit wissen wir, welche Bits invertiert werden müssen.

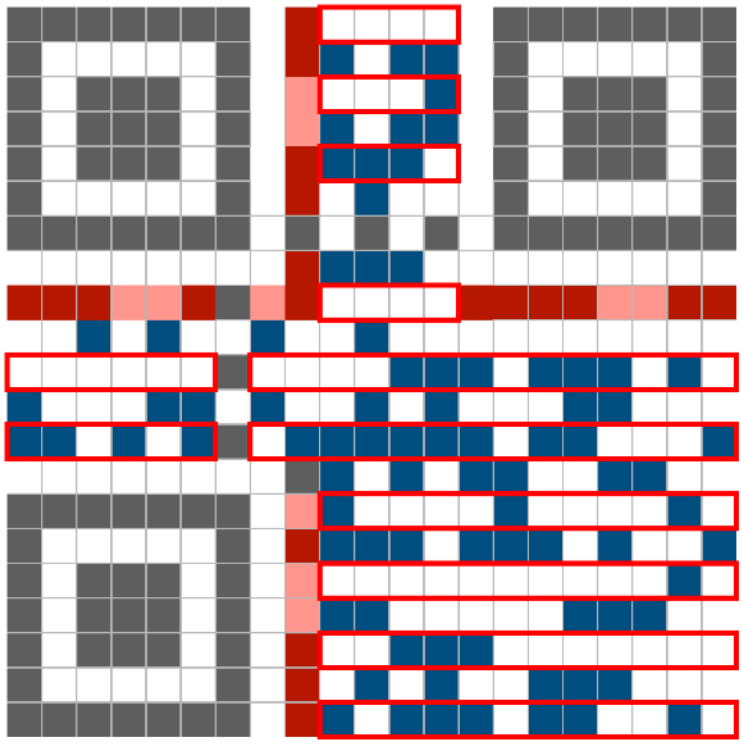
Die Maske 1 ist einfach, alle geradzahligen Zeilen (darum habe ich die auch gewählt 😉)



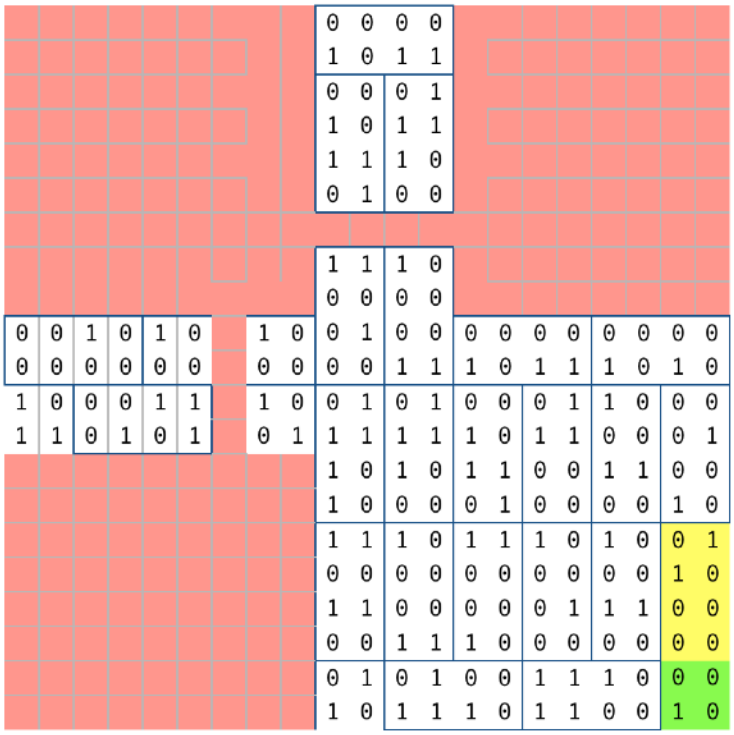
Dabei darauf achten, dass nur der Datenbereich behandelt wird. Alles, was in der oberen Grafik rot umrandet ist, muss invertiert werden. Besonders drauf achten, dass die beiden abwechselnd hell und dunkel gefärbten Timing-Marker nicht mit bearbeitet werden!

## Bits invertieren

Nach dem Invertieren der Bits sieht der Code nun so aus und wir können die Bits eintragen.



Das sieht dann so aus wie in der Abbildung unten (dunkler Block = 1, weißer Block = 0) und die Bereiche ohne Daten ausgeblendet.



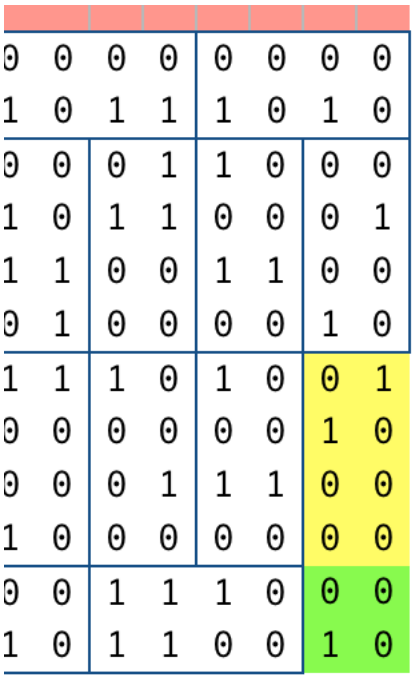
## Datentyp und Länge berechnen

Die vier Bits unten rechts tragen den Datentyp, mit der üblichen ZickZack-Leserichtung erhalten wir `0100` und damit den Code für eine byteweise Datenspeicherung.

Das oben markierte gelbe Längenfeld enthält die Bitfolge: `00000110` (dezimal 6). Damit wissen wir, dass wir 6 Bytes an Nutzlast auslesen müssen.

Die Abbildung rechts zeigt zum Nachverfolgen nochmal den rechten unteren Teil des QR-Codes. Die nächsten sechs Bytes haben die Bitfolgen:

- 0100 1000
- 0100 0001
- 0100 1100
- 0100 1100
- 0100 1111
- 0010 0001



Diese müssen nur dezimal umgerechnet und mit Hilfe einer ASCII-Tabelle in Zeichen gewandelt werden, dann haben wir das Ergebnis des QR-Codes:

```
0100 1000 = 72 = H
0100 0001 = 65 = A
0100 1100 = 76 = L
0100 1100 = 76 = L
0100 1111 = 79 = O
0010 0001 = 33 = !
```

Ergebnis: QR-Code entschlüsselt – er enthält die Zeichenfolge „HALLO!“ 👍🏼