

## 第3章 QRコードを作ろう!

### 3.1 QRコードの概要

近年、カメラ付き携帯電話の普及と共に、図 3.1 のような **QRコード**<sup>1</sup> と呼ばれる 2 次元コードをよく見かけるようになりました。QR コードは、1994 年に日本の株式会社デンソーウェーブによって開発され、1999 年には日本工業規格として「JIS X 0510 2 次元コードシンボル～QR コード～基本仕様」が制定されました。

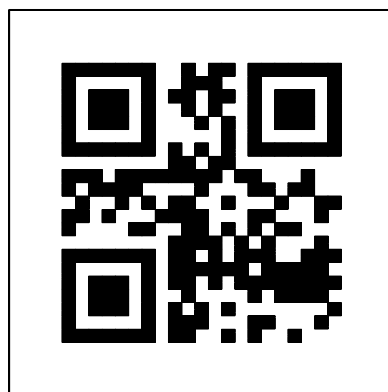


図 3.1: QR コードの例

2 次元コードの歴史と概要については、以下のホームページを参考にしてください。

- QR コードドットコム (<http://www.qrcode.com/>)  
株式会社デンソーウェーブ (<http://www.denso-wave.com/>) の運営するサイト
- まるごとわかる sensor.co.jp (<http://www.sensor.co.jp/>)  
株式会社キーエンス (<http://www.keyence.co.jp/>) の運営するサイト

QR コードの基本仕様は、以下のホームページから閲覧または購入することができます。

- 日本工業標準調査会 (<http://www.jisc.go.jp/>) 閲覧可  
JISC; Japanese Industrial Standards Committee
- 財団法人 日本規格協会 (<http://www.jsa.or.jp>) 購入可  
JSA; Japanese Standards Association

---

<sup>1</sup>この 2 次元コードが高速読み取りを重視して開発されたという経緯から、「QR コード」の“QR”は、高速読み取りを表す「クイック・レスポンス (Quick Response)」を由来としています。ただし、この 2 次元コードの正式名称は「QR コード」で、「クイック・レスポンス・コード (Quick Response Code)」の略称ではありません。なお、「QR コード」という名称は、その開発元である株式会社デンソーウェーブによって商標登録されています。

## 3.2 QRコードの作成条件と構造

以下の条件で、QRコードを作成します。

- モデル：2 (推奨されている)
- 型番：1 ( $21 \times 21$  モジュール)
- 誤り訂正レベル：L (復元能力 7%)
- マスクパターン：000 (市松模様)
- モード指示子：1000 (漢字モード)

なお、型番 1 ( $21 \times 21$  モジュール) の QR コードの構造は図 3.2 のようになっています。

- 青色 ■：位置検出パターン
- 赤色 ■：タイミングパターン
- 緑色 ■：形式情報
- 黄色 ■：データおよび誤り訂正コード語
- 白色と黒色：固定されている

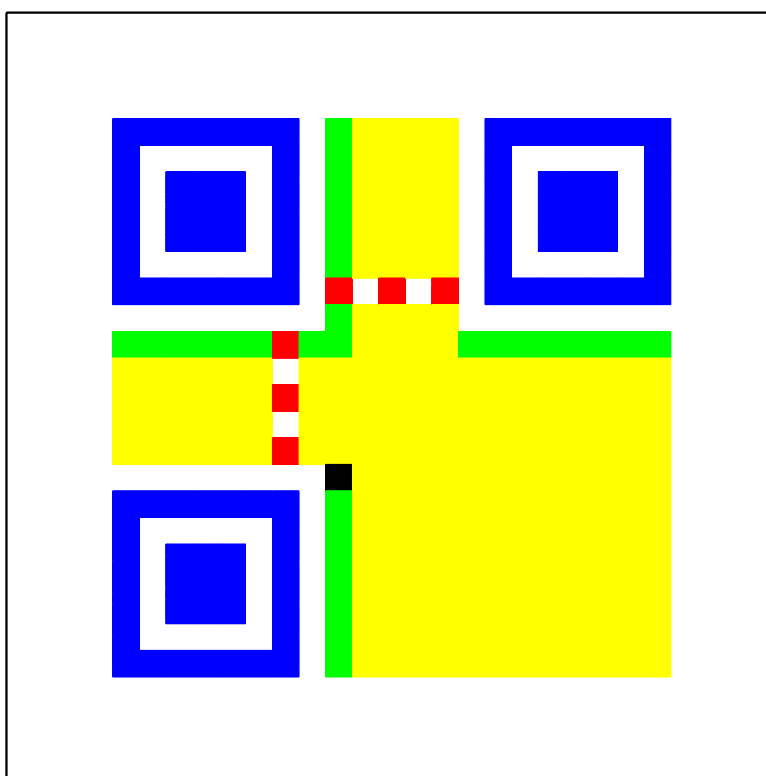


図 3.2: QRコードの構造

### 3.3 QRコードを作ろう!

作成手順は、以下のようになります<sup>2</sup>。

#### STEP1 データの符号化

- (1-1) シフト JIS 漢字コードを 13 ビットに圧縮する。
- (1-2) モード指示子 (漢字モード : 1000), 文字数指示子 (文字数を 8 ビットで表現), (1-1) で求めたデータを順に連結する。
- (1-3) (1-2) で得たデータ列に終端パターン (0000) を連結する。
- (1-4) (1-3) で得たデータ列を 8 ビットごとに区切る。  
なお、残りビットが 8 ビットに満たなければ 0 で埋めて 8 ビットにする。
- (1-5) (1-4) で得たデータ列に、データ容量を満たすまで、埋め草コード語 11101100 および 00010001 を交互に付加する。
- (1-6) (1-5) で得たデータ列から誤り訂正コード語 (生成多項式による剰余) を求める。
- (1-7) (1-5) で得たデータ列に (1-6) で得たデータ列を連結する。

#### STEP2 マスク処理

- (2-1) 仕様にしたがって、STEP1 の (1-7) で得たデータ列を QR コードのデータおよび誤り訂正コード語領域に配置する (= モジュールパターン)。
- (2-2) モジュールパターンに市松模様 (000) のマスク処理を施す。

#### STEP3 形式情報の符号化

- (3-1) 誤り訂正レベル (L : 01) とマスクパターン (市松模様 : 000) を連結する。
- (3-2) (3-1) で得たデータ列 (01000) から誤り訂正コード語 (生成多項式による剰余) を求める。
- (3-3) (3-1) で得たデータ列に (3-2) で得たデータ列を連結する。
- (3-4) (3-3) で得たデータ列と 1010100000010010 の排他的論理和をとる。
- (3-5) 仕様にしたがって、(3-4) で得たデータ列を形式情報領域に配置する。

**完成!!!**

では、実際に QR コードを作成して行きましょう。例として、漢字文字列「幸山直人」を QR コードにして行きます。

---

<sup>2</sup>作成作業を容易にするために、幾つかの手順が簡略化されています。

**STEP1** データの符号化(1-1) シフト JIS 漢字コードを 13 ビットに圧縮する。

まず、各漢字 ”幸”, ”山”, ”直”, ”人” を対応するシフト JIS 漢字コードに変換します。

$$( \text{幸}, \text{山}, \text{直}, \text{人} ) \rightarrow ( 8D4B_{16}, 8E52_{16}, 92BC_{16}, 906C_{16} )$$

次に、各シフト JIS 漢字コードを 13 ビットに圧縮します。8140<sub>16</sub>～9FFC<sub>16</sub> であれば 8140<sub>16</sub> を、E040<sub>16</sub>～EBBF<sub>16</sub> であれば C140<sub>16</sub> を、それぞれ減じます。

$$\begin{aligned} &\rightarrow ( 8D4B_{16} - 8140_{16}, 8E52_{16} - 8140_{16}, 92BC_{16} - 8140_{16}, 906C_{16} - 8140_{16} ) \\ &\rightarrow ( 0C0B_{16}, 0D12_{16}, 117C_{16}, 0F2C_{16} ) \end{aligned}$$

さらに、上位 2 バイトに C0<sub>16</sub> を乗じ、下位 2 バイトを加算します。

$$\begin{aligned} &\rightarrow ( 0C_{16} \times C0_{16} + 0B_{16}, 0D_{16} \times C0_{16} + 12_{16}, 11_{16} \times C0_{16} + 7C_{16}, 0F_{16} \times C0_{16} + 2C_{16} ) \\ &\rightarrow ( 090B_{16}, 09D2_{16}, 0D3C_{16}, 0B6C_{16} ) \end{aligned}$$

13 ビット以下で表現可能な数となっているので、それぞれ 13 ビットの 2 進数に変換します。

$$\rightarrow ( 0100100001011_2, 0100111010010_2, 0110100111100_2, 0101101101100_2 )$$

(1-2) モード指示子 (漢字モード : 1000), 文字数指示子 (文字数を 8 ビットで表現), (1-1) で求めたデータを順に連結する。

漢字 4 文字なので、文字数指示子は 00000100 となります。モード指示子, 文字数指示子, (1-1) で求めたデータを順に連結します。

$$\rightarrow ( \text{1000}, \text{00000100}, 0100100001011, 0100111010010, 0110100111100, 0101101101100 )$$

注意 : 3 文字の人は 00000011 が、5 文字の人は 00000101 が、文字数指示子となります。

(1-3) (1-2) で得たデータ列に終端パターン (0000) を連結する。

(1-2) で得たデータ列に終端パターン (0000) を連結します。

$$\rightarrow ( 1000, 00000100, 0100100001011, 0100111010010, 0110100111100, 0101101101100, \text{0000} )$$

(1-4) (1-3) で得たデータ列を 8 ビットごとに区切る。

(1-3) で得たデータ列を 8 ビットごとに区切り直します。最後は、8 ビットに満たないので 0000 を付け加えて、8 ビット長にします。

$$\rightarrow ( 10000000, 01000100, 10000101, 10100111, 01001001, 10100111, 10001011, 01101100, 0000\text{0000} )$$

注意 : 3 文字の人は 0 を、5 文字の人は 00000000 を、それぞれ終端に補って 8 ビット長にします。

(1-5) (1-4) で得たデータ列に、データ容量を満たすまで、埋め草コード語 11101100 および 00010001 を交互に付加する。

19 バイトのデータ長になるように、埋め草コード語 11101100 および 00010001 を交互に付加します。

→ ( 10000000, 01000100, 10000101, 10100111, 01001001,  
10100111, 10001011, 01101100, 00000000, 11101100,  
00010001, 11101100, 00010001, 11101100, 00010001,  
11101100, 00010001, 11101100, 00010001 )

(1-6) (1-5) で得たデータ列から誤り訂正コード語 (生成多項式による剰余) を求める。

$GF(2^8)$  上の 3 個 (最小距離 8) の誤りが訂正可能な [26,19]RS 符号として、誤り訂正コード語を求めます ( $q = 2^8$ ,  $m = 1$ ,  $t = 3$ )。すなわち、 $GF(2^8)$  の原始元を  $\alpha$  とすると、生成多項式は

$$\begin{aligned} G(x) &= (x - \alpha^0)(x - \alpha^1)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)(x - \alpha^5)(x - \alpha^6) \\ &= x^7 + \alpha^{87}x^6 + \alpha^{229}x^5 + \alpha^{146}x^4 + \alpha^{149}x^3 + \alpha^{238}x^2 + \alpha^{102}x + \alpha^{21} \end{aligned}$$

となります。また、(1-5) で得たデータ列を  $GF(2^8)$  の元で表せば (ベクトル表現→べき表現)、

→ (  $\alpha^7$ ,  $\alpha^{102}$ ,  $\alpha^{128}$ ,  $\alpha^{205}$ ,  $\alpha^{152}$ ,  $\alpha^{205}$ ,  $\alpha^{237}$ ,  $\alpha^{250}$ , 0,  $\alpha^{122}$ ,  
 $\alpha^{100}$ ,  $\alpha^{122}$ ,  $\alpha^{100}$ ,  $\alpha^{122}$ ,  $\alpha^{100}$ ,  $\alpha^{122}$ ,  $\alpha^{100}$ ,  $\alpha^{122}$ ,  $\alpha^{100}$  )

となるので、データ (情報) を多項式表現すると

$$\begin{aligned} I(x) &= \alpha^7 x^{18} + \alpha^{102} x^{17} + \alpha^{128} x^{16} + \alpha^{205} x^{15} + \alpha^{152} x^{14} + \alpha^{205} x^{13} \\ &\quad + \alpha^{237} x^{12} + \alpha^{250} x^{11} + 0x^{10} + \alpha^{122} x^9 + \alpha^{100} x^8 + \alpha^{122} x^7 \\ &\quad + \alpha^{100} x^6 + \alpha^{122} x^5 + \alpha^{100} x^4 + \alpha^{122} x^3 + \alpha^{100} x^2 + \alpha^{122} x + \alpha^{100} \end{aligned}$$

となります。したがって、誤り訂正コード語の多項式表現は

$$\begin{aligned} R(x) &= [I(x)x^7] \bmod G(x) \\ &= \alpha^{214} x^6 + \alpha^{83} x^5 + \alpha^{238} x^4 + \alpha^{63} x^3 + \alpha^{179} x^2 + \alpha^{221} x + \alpha^{230} \end{aligned}$$

となります ( $7 = 26 - 19$ )。

注意：本来、 $GF(2^8)$  上の RS 符号の符号長は 254 ( $= (2^8 - 1) - 1$ ) ですが、下記のようにデータの high の係数を全て 0 とみなすことで符号長を短縮しています。

$$(0, 0, \dots, 0, 0, x_{25}, x_{24}, \dots, x_1, x_0).$$

さらに、生成多項式についても、誤り訂正可能な数が 3 個であることから最小多項式の次数は 6 でよいのですが、7 にしても最小距離が 1 増えるだけなので誤り訂正可能な数は変化しません。このとき、QR コードの復号誤りの可能性を低減するための条件を加えると、実質的な誤り訂正可能な数は 2 個となります (理論上の誤り訂正可能な数は 3 個)。したがって、符号長 26 のうち 2 個の誤りが訂正可能な誤り訂正符号となり、誤り訂正率は  $2/26 = 0.0769$  となります (すなわち、誤り訂正レベル L (7%以上) を満たす)。

(1-7) (1-5) で得たデータ列に (1-6) で得たデータ列を連結する。

符号語 (データ + 誤り訂正データ語) を作成しましょう。GF( $2^8$ ) 上および GF(2) 上の演算であることに注意すれば、巡回符号 (RS 符号) の符号語の多項式表現は

$$X(x) = I(x)x^7 - R(x) = I(x)x^7 + R(x)$$

によって与えられましたから、(1-6) より多項式の係数を並べれば、符号語

$$\rightarrow (\alpha^7, \alpha^{102}, \alpha^{128}, \alpha^{205}, \alpha^{152}, \alpha^{205}, \alpha^{237}, \alpha^{250}, 0, \alpha^{122}, \alpha^{100}, \alpha^{122}, \alpha^{100}, \alpha^{122}, \alpha^{100}, \alpha^{122}, \alpha^{100}, \alpha^{122}, \alpha^{100}, \alpha^{214}, \alpha^{83}, \alpha^{238}, \alpha^{63}, \alpha^{179}, \alpha^{221}, \alpha^{230})$$

が得られます。これをベクトル表現すれば、データおよび誤り訂正コード語領域を埋める 208 (=  $26 \times 8$ ) 個の 2 元符号が得られます：

$$\rightarrow (10000000, 01000100, 10000101, 10100111, 01001001, 10100111, 10001011, 01101100, 00000000, 11101100, 00010001, 11101100, 00010001, 11101100, 00010001, 11101100, 00010001, 11101100, 00010001, 11111001, 10111011, 00001011, 10100001, 01001011, 01000101, 11110100)$$

注意：第2章では右から高次の係数を並べましたが、ここでは、左から高次の係数を並べます。これは本質的な問題ではなく、仕様の問題です。