QR コードはモジュールと呼ばれる正方形の碁盤の目に白黒を配置することにより生成される 2 次元バーコードである．1 つのモジュールは，特徴的な 3 つの目玉のような位置検出パターンや，1 マスの大きさを示すタイミングパターンなどの読み取りのための**機能パターン**と，情報語を RS 符号によって符号化したものと，復号に必要な形式情報を合わせた**符号領域**からなっている．現在 QR コードは JIS 規格や国際規格で規格化されており，JIS 規格はインターネットサイト（https://kikakurui.com/x0/X0510-2018-01.html）で簡易閲覧可能になっている．

ここでは，個人のサイト（https://www.swetake.com/qrcode/qr1.html）を参考にして学籍番号を実際に QR コードにしてみる．（「QR コードを目で読む」と銘打った次のサイトもいろいろ参考になると思う．https://ameblo.jp/neco-home/entry-12592290330.html ）

● QR コードの型

中大の学籍番号は大文字のみのアルファベットと数字あわせて 11 文字からなるので「英数字モード」を用いることにする．この文字数なら誤り訂正レベルが高めの「レベル Q」を選択しても 1 型（21 × 21 セル）に収まるので，1-Q 型を選ぶことにする．また，マスクパターンは本来出来上がったデータの上に掛けてみて一番良いものを選ぶべきなのだが，ここではそれを省略し予め市松模様のパターン（000 型）をかけることに決めておく．

型番（バージョン）： 1 （21 × 21 セル）
誤り訂正レベル： Q（2 進指示子 11，復元能力 25%）
マスクパターン： 000（市松模様）
データモード： 0010（英数字モード）

● 形式情報

誤り訂正レベルとマスクパターンは「形式情報」としてタイミングパターンのすぐそばに格納される．形式情報は誤り訂正レベル 2bit と，マスクパターン 5bit を BCH(15, 5) で符号化し，さらに 101010000010010 というマスクを掛けてつくられる．BCH (15, 5) は 5bit の情報語に 10bit の検査 bit を加え 15bit を送受信する．BCH(15, 5) は $\mathbb{F}_{16} = \mathbb{F}_2[x]/(x^4 + x + 1) = \mathbb{F}_2(\beta)$ を利用した符号で，生成多項式 $g(x)$ を

$$\begin{aligned} g(x) &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) \\ &= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \end{aligned}$$

とする．これは，$g(\beta) = g(\beta^3) = g(\beta^5) = 0$ となる一番次数の低い多項式である．5bit の情報語を 4 次の情報多項式 $q(x)$ に変換し，生成多項式 $g(x)$ を用いて，送信多項式 $u(x)$ を次のようにする．

$$\text{送信多項式} : u(x) = q(x)x^{10} + \big(q(x)x^{10} \text{ を } g(x) \text{ で割った余り}\big)$$

さらに，この結果にマスク 101010000010010 をかけ，0 または 1 が過度に連続することを防ぐ．我々の場合の 11000 からえら得る符号は次のようになる．

011010101011111

入学年度		学部	学 科		組		番　号			検	フリガナ	
		D		1							氏 名	

● データの bit 列化

さて，次にデータ領域に配置するデータを準備する．まず，最初の 4bit はモードを指示するためのもので，学籍番号では英数字モードを用いるので 0010 とする．次に文字数を指示する必要があるが，英数字モードの場合の最大格納文字数の関係から，これを 9bit で表す．中大の学籍番号は 11 文字なので，これを 2 進法で表示すると 1011 であるが，これに 0 を加えて 9bit 化し 000001011 とする．

そして，いよいよ実際のデータを bit 列になおす．英数字モードではまず下表の通りに各文字を数字化する．なぜ 45 文字が使用可能かというと，$45^2 = 2025 \fallingdotseq 2048 = 2^{11}$ なので，2 文字の組を 11bit で表すことができ，効率よく符号化できるからである．そこで，データを 2 文字ずつに区切り，1 つ目の文字の下の表の値を 45 倍したものと 2 つ目の文字の表の値を足す．（2 文字の並びは「45 進法」で表されていると考え，これを 10 進法に直すことに相当する．）

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		$	%	*
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
+	-	.	/	:															
40	41	42	43	44															

たとえば，“22” という 2 文字は $2 \times 45 + 2 = 92$ という 10 進数で表され，“D1” は $13 \times 45 + 1 = 586$ となる．なお文字数が奇数の場合は最後に残った 1 文字は対応する値をそのままの値とする．さらに，こうして計算された数を 11bit の 2 進数で表す．たとえば，“92” は 2 進法で 1011100 だが，これを 11bit にするために最初に 0 をいくつか加え， 00001011100 とする．以下，これをこれを続け，文字数が奇数の場合の最後に残った 1 文字は対応する値を 2 進法で表し，6 bit で表記する．

学生証番号	2	2	D								
「10 進法化」	92										
11bit 化	00001011100										

すべてを bit 化したら，最後に終端パターンとして 0000 を付加する．

$$\underbrace{0010}_{\text{英数字モード}}\ \underbrace{000001011}_{\text{文字数 11}}\underbrace{00001011100}_{\text{“22”}}\underbrace{\cdots}_{\text{“D・”}}\ \cdots\ \underbrace{0000}_{\text{終端パターン}}$$

こうして得られたデータを 8bit ごと（1byte ごと）に区切り直す．最後のビット列が 8bit 未満の場合は 0 で埋める．また，1-Q 型では RS(26, 13) 符号を用いるので，得られた byte 数が情報 byte 数である 13 に満たない場合は "11101100" および "00010001" という「埋め草パターン」を交互に付加しする．さらに，これを 8bit=1byte ごとに分割し，13byte のデータとする．裏面のデータ表を完成させよ．

● 誤り訂正符号化

さて，こうしてできた 13byte の情報語から RS(26, 13) と呼ばれる RS 符号（リード・ソロモン符号）により誤り訂正コード語を加えて符号化する．

【裏に続く】

英数字モード				文字数 11								“22”											
0	0	1	0	0	0	0	0	0	1	0	1	1											
“D . ”																							
		終端パターン				0 fill		埋め草パターン 1								埋め草パターン 2							
		0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	1
埋め草パターン 1																							
1	1	1	0	1	1	0	0																

これを 8bit=1byte ごとに分割し，13byte からなる情報語を得る．

	8bit データ							
1.	0	0	1	0	0	0	0	0
2.	0	1	0	1	1	0	0	0
3.								
4.								
5.								
6.								
7.								
8.								
9.								
10.								
11.								
12.								
13.	1	1	1	0	1	1	0	0

RS 符号は $GF(2^8) = GF(256) = \mathbb{F}_{256}$ という数の体系を用いて作られる．$\mathbb{F}_{256}$ は $GF(2) = \mathbb{F}_2$ に

$$\gamma^8 + \gamma^4 + \gamma^3 + \gamma^2 + 1 = 0$$

をみたす γ という “虚数” を付け加えた数の体系である．$\mathbb{F}_{256}$ の数は γ の 7 次以下の多項式で表され（加法表示），8 bit = 1 byte の情報を保持する．また，$\mathbb{F}_{256}$ の 0 以外の数は γ^k $(k = 0, 1, \ldots, 254)$ と表せることも思い出しておく（乗法表示）．

これまで作成されたデータは 13 byte あるが，その各 byte を γ の 7 次以下の多項式とみなし，$\mathbb{F}_{256}$ の元とみなす．たとえば，1 行目の “00100000” は γ^5，2 行目の “01011000” は $\gamma^6 + \gamma^4 + \gamma^3$ などとする．そして，この 13 byte の情報語から，係数が $GF(2^8)$ の元である x の 12 次の多項式をつくる．こうして情報語から

$$q(x) = \gamma^5 x^{12} + (\gamma^6 + \gamma^4 + \gamma^3)x^{11} + \cdots + (\gamma^7 + \gamma^6 + \gamma^5 + \gamma^3 + \gamma^2)$$

という情報多項式が得られる．

1 先に作成した 13byte の情報語から，情報多項式 $q(x)$ をつくれ．

$$\begin{aligned} q(x) = (&\ )x^{12} \\ +(&\ )x^{11} \\ +(&\ )x^{10} \\ +(&\ )x^{9} \\ +(&\ )x^{8} \\ +(&\ )x^{7} \\ +(&\ )x^{6} \\ +(&\ )x^{5} \\ +(&\ )x^{4} \\ +(&\ )x^{3} \\ +(&\ )x^{2} \\ +(&\ )x \\ +(&\ ) \end{aligned}$$

BCH 符号では，情報多項式 $q(x)$ から生成多項式 $g(x)$ を用いて送信多項式を作るのであった．RS 符号でも，BCH 符号と同じ要領で送信多項式を作る．RS(26, 13) では，生成多項式 $g(x)$ を

$$g(x) = (x-1)(x-\gamma)(x-\gamma^2)(x-\gamma^3) \times \cdots \times (x-\gamma^{12})$$

として，送信多項式 $u(x)$ を $g(x)$ 用いて次のようにする．

$$u(x) = q(x)x^{13} + \big(q(x)x^{13} \text{ を } g(x) \text{ で割った余り}\big)$$

次回，Mathematica を用いて $u(x)$ を計算し，それを元に QR コードを作ってみることにする．

担当：鍬田 政人 (kuwata.04f@g.chuo-u.ac.jp)